

AUTOMATED INTERACTIVE SIMULATION MODEL (AISM) VAX
VERSION 30 USER'S MANUAL (U) HUGHES AIRCRAFT CO
FULLERTON CA GROUND SYSTEMS GROUP V ALLERTON ET AL.
29 MAY 87 ESD-TR-87-228 F19628-06-C-0070 F/8 12/5

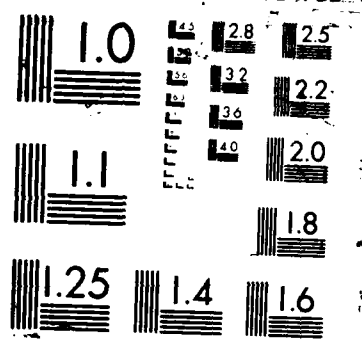
14

UNCLASSIFIED

F/8 12/5

風

The image displays a 10x10 grid of 100 small, square images. Each image appears to be a scan of a document page, showing various elements such as text, diagrams, and tables. The images are arranged in a regular grid pattern, with some images showing more content than others. The overall appearance is that of a digital archive or a collection of document snippets.



DTIC FILE COPY

ESD-TR-87-228

CDRL 105

4

Automated Interactive Simulation Model (AISIM)
VAX Version 5.0 User's Manual

VICKY ALLERTON
GLORIA BOICE
SUSAN SWEET

Hughes Aircraft Company
Ground Systems Group
P.O. Box 3310
Fullerton, CA 92634

29 May 1987

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
DEC 31 1987
S H D

Prepared For

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY FOR DEVELOPMENT PLANS AND SUPPORT SYSTEMS
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731



AD-A190 171

87 12 21 117


LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

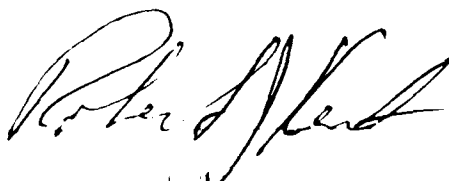
This report has been reviewed and is approved for publication.



William H. D. [unclear]
[unclear]
[unclear]



Charles J. Ryan, Major, USAF
Program Manager, Computer Resource
Management Technology Program
AFM 64-401



Robert J. [unclear]
[unclear]
[unclear]

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified			1b RESTRICTIVE MARKINGS AD-190171	
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE				
4 PERFORMING ORGANIZATION REPORT NUMBER(S) CDRL 105 1854895-2			5 MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-87-228	
6a NAME OF PERFORMING ORGANIZATION Hughes Aircraft Company Ground Systems Group	6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION HQ, Electronic Systems Division (XRSE)		
6c ADDRESS (City, State, and ZIP Code) P.O. Box 3310 Fullerton, CA 92634		7b ADDRESS (City, State, and ZIP Code) Hanscom AFB Massachusetts, 01731-5000		
8a NAME OF FUNDING/SPONSORING ORGANIZATION Electronic Systems Division	8b OFFICE SYMBOL (If applicable) XRSE	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-86-C-0070		
8c ADDRESS (City, State, and ZIP Code) Hanscom AFB Massachusetts, 01731-5000		10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO	PROJECT NO	TASK NO
		WORK UNIT ACCESSION NO		
11 TITLE (Include Security Classification) Automated Interactive Simulation Model (AISIM) VAX Version 5.0 User's Manual				
12 PERSONAL AUTHOR(S) Vicky Allerton, Gloria Boice, Susan Sweet				
13a TYPE OF REPORT Final	13b TIME COVERED FROM 5/14/86 TO 5/15/87	14 DATE OF REPORT (Year, Month, Day) 1987 May 29	15 PAGE COUNT 372	
16 SUPPLEMENTARY NOTATION				
17 COSATI CODES FIELD GROUP SUB-GROUP			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) AISIM Acceptance Test Test Procedures	
19 ABSTRACT (Continue on reverse if necessary and identify by block number) This document is the User's Manual for the Hughes developed Automated Interactive Simulation Model (AISIM) Version 5.0 for VAX 11/780 hosts. This manual provides the user with a comprehensive guide for using this system to perform high level simulation of operational and distributed data processing systems.				
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Michael Merriman, 1Lt, USAF			22b TELEPHONE (Include Area Code) (617) 377-2716	22c OFFICE SYMBOL XRSE

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

Section	Page
1. INTRODUCTION	1-1
1.1 PURPOSE AND SCOPE	1-1
1.2 ORGANIZATION	1-1
1.3 DOCUMENTATION CONVENTIONS	1-1
1.4 APPLICABLE DOCUMENTS	1-3
2. AISIM CONCEPTS	2-1
2.1 CHARACTERISTICS OF SYSTEMS MODELED BY AISIM	2-1
2.2 MODELING	2-2
2.3 DESIGNING MODELS	2-2
2.4 CONSTRUCTING AN AISIM MODEL	2-3
2.4.1 Charting a Paper Model	2-3
2.4.2 Defining the AISIM Model	2-4
2.5 AISIM MODELING ENTITIES	2-4
3. AISIM ENTITIES AND OTHER MODELING CONSTRUCTS	3-1
3.1 SCENARIO	3-2
3.2 LOAD	3-4
3.3 ITEM	3-7
3.4 USER DEFINED QUEUES	3-9
3.5 SYSTEM DEFINED QUEUES	3-11
3.5.1 States Associated with Resources	3-11
3.5.2 Cross Reference Sets	3-12
3.6 RESOURCE	3-13
3.7 ACTION	3-16
3.8 PROCESS	3-17
3.9 PRIMITIVES	3-21
3.9.1 ACTION	3-24
3.9.2 ALLOC	3-26
3.9.3 ASSIGN	3-27
3.9.4 BRANCH	3-29
3.9.5 CALL	3-30
3.9.6 COMMENT	3-32
3.9.7 COMPARE	3-33
3.9.8 CREATE	3-35
3.9.9 DEALLOC	3-36
3.9.10 DESTROY	3-37
3.9.11 ENTRY	3-38
3.9.12 EVAL	3-39
3.9.13 FILE	3-43
3.9.14 FIND	3-44
3.9.15 LOCK	3-45
3.9.16 LOOP	3-46
3.9.17 PROB	3-47
3.9.18 READ	3-48
3.9.19 REMOVE	3-50
3.9.20 RESET	3-51
3.9.21 RESUME	3-52
3.9.22 SEND	3-53

Mission For	
GRA&I	<input checked="" type="checkbox"/>
TAB	<input type="checkbox"/>
ounced	<input type="checkbox"/>
ification	
tribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

3.9.23	SUSPEND	3-54
3.9.24	TEST	3-55
3.9.25	TRACE	3-56
3.9.26	UNLOCK	3-58
3.9.27	WAIT	3-59
3.9.28	WRITE	3-60
3.10	LEGAL PATH TABLE - NODE - LINK	3-62
3.11	FILES	3-64
3.12	TABLES	3-66
3.12.1	Discrete Tables	3-66
3.12.2	Continuous Tables	3-66
3.12.3	Alphanumeric Tables	3-66
3.13	ATTRIBUTES	3-68
3.14	CONSTANTS AND GLOBAL VARIABLES	3-69
3.15	LOCAL VARIABLES	3-71
3.16	ALPHA LITERALS	3-73
3.17	KEYWORDS	3-74
3.18	MESSAGE ROUTING SUBMODEL	3-76
4.	AIMS SYSTEM OVERVIEW AND SYSTEM INITIALIZATION	4-1
4.1	REACHING THE AIMS READY LEVEL	4-3
4.2	ACCESSING AIMS HELP	4-4
5.	AIMS READY LEVEL	5-1
5.1	INITIATING AN ANALYSIS SESSION	5-4
5.2	BACKING UP A DATABASE	5-6
5.3	RUNNING AN ANALYZE SESSION VIA BATCH MODE	5-7
5.4	CHANGING THE CURRENT PARAMETERS	5-10
5.5	DELETING PROJECT FILES	5-11
5.6	INITIATING A DESIGN SESSION	5-12
5.7	VIEWING OUTPUT REPORTS	5-13
5.8	RETURNING TO VAX/VMS READY LEVEL	5-14
5.9	CREATING AND EDITING AN INPUT FILE FOR THE READ PRIMITIVE	5-15
5.10	CREATING A MODEL LISTING	5-16
5.11	HARDCOPY OUTPUT OF THE PROCESS FLOWCHARTS	5-18
5.12	OBTAINING HELP FROM THE SYSTEM	5-19
5.13	INITIALIZING A HELP EDITOR SESSION	5-21
5.14	EXERCISING THE LIBRARY FACILITY	5-22
5.15	LISTING THE CURRENT OPTIONS	5-23
5.16	LISTING THE COMMAND PROCEDURE LINES	5-24
5.17	DISABLE THE LISTON OPTION	5-25
5.18	DISABLE AIMS MESSAGES	5-26
5.19	DISABLE MSGOFF FEATURE	5-27
5.20	PRINTING OUTPUT REPORTS	5-28
5.21	INITIATING A REplot SESSION	5-29
5.22	RESTORING A DATABASE (AFTER A CATASTROPHE HAS OCCURRED)	5-30
6.	DESIGN USER INTERFACE (DUI)	6-1
6.1	DUI COMMAND SUMMARY	6-5
6.1.1	DUI COMMAND: ARCH	6-6
6.1.2	DUI COMMAND: COPY	6-7
6.1.3	DUI COMMAND: DELETE	6-8

6.1.4	DUI COMMAND: EDIT	6-9
6.1.5	DUI COMMAND: END	6-10
6.1.6	DUI COMMAND: HELP	6-11
6.1.7	DUI COMMAND: LIST	6-13
6.1.8	DUI COMMAND: SAVE	6-14
6.1.9	DUI COMMAND: UNITS	6-15
6.1.10	Termination of a DUI Session	6-16
6.2	PROCESS EDITOR INTERFACE (PEI)	6-17
6.2.1	Use of the PEI	6-17
6.2.2	PEI COMMAND: BOTTOM	6-20
6.2.3	PEI COMMAND: CHANGE	6-21
6.2.4	PEI COMMAND: DELETE	6-22
6.2.5	PEI COMMAND: DOWN	6-23
6.2.6	PEI COMMAND: DRAW	6-24
6.2.7	PEI COMMAND: END	6-25
6.2.8	PEI COMMAND: HELP	6-26
6.2.9	PEI COMMAND: HOLD	6-28
6.2.10	PEI COMMAND: MENU	6-29
6.2.11	PEI COMMAND: NODRAW	6-30
6.2.12	PEI COMMAND: PLACE	6-31
6.2.13	PEI COMMAND: REDRAW	6-32
6.2.14	PEI COMMAND: TOP	6-33
6.2.15	PEI COMMAND: UP	6-34
6.2.16	Terminating a PEI Session	6-35
6.3	ARCHITECTURE DESIGN EDITOR (ADE)	6-36
6.3.1	Concepts For Using ADE	6-36
6.3.2	Use of the ADE	6-38
6.3.3	ADE Symbols	6-40
6.3.4	ADE COMMAND: CHANGE	6-41
6.3.5	ADE COMMAND: CONNECT	6-42
6.3.6	ADE COMMAND: DEFINE	6-44
6.3.7	ADE COMMAND: DELETE	6-47
6.3.8	ADE COMMAND: DRAW	6-49
6.3.9	ADE COMMAND: END	6-49
6.3.10	ADE COMMAND: HELP	6-50
6.3.11	ADE COMMAND: LIST	6-52
6.3.12	ADE COMMAND: MOVE	6-53
6.3.13	ADE COMMAND: NODRAW	6-54
6.3.14	ADE COMMAND: PLACE	6-55
6.3.15	ADE COMMAND: RECON	6-56
6.3.16	ADE COMMAND: REDRAW	6-57
6.3.17	ADE COMMAND: SAVE	6-58
6.3.18	ADE COMMAND: WINDOW	6-59
6.3.19	Termination of an ADE Session	6-60
7.	ANALYSIS USER INTERFACE (AUI)	7-1
7.1	AUI COMMAND: CANBREAK	7-7
7.2	AUI COMMAND: DEFLOT	7-8
7.3	AUI COMMAND: DELETE	7-12
7.4	AUI COMMAND: EDIT	7-13
7.5	AUI COMMAND: END	7-14
7.6	AUI COMMAND: GET	7-15
7.7	AUI COMMAND: GO	7-16
7.8	AUI COMMAND: HELP	7-17

7.9	AUI COMMAND: INFRES	7-19
7.10	AUI COMMAND: LIST	7-20
7.11	AUI COMMAND: LISTVAL	7-21
7.12	AUI COMMAND: PLOT	7-22
7.13	AUI COMMAND: SAVE	7-24
7.14	AUI COMMAND: SETBREAK	7-25
7.15	AUI COMMAND: UNITS	7-26
7.16	TERMINATION OF AN AUI SESSION	7-27
8.	REPLOT USER INTERFACE (RUI)	8-1
8.1	RUI COMMAND: CLEAR	8-3
8.2	RUI COMMAND: DELETE	8-4
8.3	RUI COMMAND: END	8-5
8.4	RUI COMMAND: GET	8-6
8.5	RUI COMMAND: HELP	8-7
8.6	RUI COMMAND: LIST	8-9
8.7	RUI COMMAND: PLOT	8-10
8.8	RUI COMMAND: SAVE	8-11
8.9	RUI COMMAND: UNITS	8-12
9.	HARDCOPY USER INTERFACE (HUI)	9-1
10.	LIBRARY USER INTERFACE (LUI)	10-1
10.1	LUI COMMAND: CHECKIN	10-4
10.2	LUI COMMAND: CHECKOUT	10-5
10.2.1	CO COMMAND: DELETE	10-7
10.2.2	CO COMMAND: END	10-8
10.2.3	CO COMMAND: EXTRACT	10-9
10.2.4	CO COMMAND: HELP	10-10
10.2.5	CO COMMAND: LIST	10-12
10.3	LUI COMMAND: CONVERT	10-13
10.4	LUI COMMAND: HELP	10-14
10.5	LUI COMMAND: MERGEIN	10-16
10.5.1	MI COMMAND: END	10-13
10.5.2	MI COMMAND: HELP	10-19
10.5.3	MI COMMAND: IGNORE	10-21
10.5.4	MI COMMAND: INFO	10-22
10.5.5	MI COMMAND: RENAME	10-23
10.5.6	MI COMMAND: REPLACE	10-24
10.6	LUI COMMAND: MERGEOUT	10-25
10.6.1	MO COMMAND: END	10-27
10.6.2	MO COMMAND: HELP	10-28
10.6.3	MO COMMAND: LIST	10-30
10.6.4	MO COMMAND: SELECT	10-31
11.	HELP EDITOR INTERFACE	11-1
11.1	HEI COMMAND SUMMARY	11-2
11.1.1	HEI COMMAND: END	11-3
11.1.2	HEI COMMAND: HELP	11-4
11.1.3	HEI COMMAND: UPDATE	11-6
11.2	UPDATE (UPD)	11-7
11.2.1	Update Command Summary	11-8
11.2.2	UPDATE COMMAND: ADD	11-9
11.2.3	UPDATE COMMAND: CHANGE	11-10

11.2.4	UPDATE COMMAND: DELETE	11-11
11.2.5	UPDATE COMMAND: END	11-12
11.2.6	UPDATE COMMAND: HELP	11-13
11.2.7	UPDATE COMMAND: LIST	11-15
11.2.8	UPDATE COMMAND: SAVE	11-16
12.	FILE MANAGEMENT USER INTERFACE (FUI)	12-1
12.1	FUI COMMAND SUMMARY	12-3
12.1.1	FUI COMMAND: DELETE	12-4
12.1.2	FUI COMMAND: END	12-5
12.1.3	FUI COMMAND: HELP	12-6
12.1.4	FUI COMMAND: LIST	12-8
12.1.5	FUI COMMAND: LISTOFF	12-9
12.1.6	FUI COMMAND: LISTON	12-10
12.1.7	FUI COMMAND: PLACE	12-11
12.1.8	FUI COMMAND: RENUM	12-12
13.	AISIM SIMULATION REPORTS	13-1
13.1	INTERACTIVE RESULTS AND HOW TO OBTAIN THEM	13-1
13.2	REPORT RESULTS AND HOW TO OBTAIN THEM	13-2
13.2.1	Constant Report	13-8
13.2.2	Variable Report	13-9
13.2.3	Item Report	13-11
13.2.4	Resource Report	13-12
13.2.5	Action Report	13-14
13.2.6	Queue Report	13-16
13.2.7	Process Report	13-18
13.3	COMMANDS RELEVANT TO VIEWING OUTPUT REPORTS	13-21
13.3.1	TOP, BOTTOM	13-21
13.3.2	UP, DOWN	13-21
13.3.3	FIND	13-21
13.3.4	LIST	13-22
APPENDIX A	OPERATIONAL PROCEDURES AND IMPORTANT INFORMATION	A-1
A.1	IMPORTANCE OF DATABASE BACKUP AND ALLOCATION	A-1
A.2	ABNORMAL TERMINATION OF A DUI OR AUI SESSION	A-1
A.3	AISIM PLOTS	A-2
A.4	PRODUCING HARDCOPIES OF THE TERMINAL DISPLAY	A-3
A.5	RANDOMNESS IN RESULTS	A-5
APPENDIX B	AISIM ERRORS	B-1
APPENDIX C	GLOSSARY	C-1
APPENDIX D	MESSAGE ROUTING SUBMODEL	D-1

LIST OF ILLUSTRATIONS

FIGURE	PAGE
2-1 AISIM Entity Relationships	2-4
3-1 Form for the Scenario Entity	3-2
3-2 Form for the Load Entity	3-4
3-3 Form for the Item Entity	3-7
3-4 Form for the Queue Entity	3-10
3-5 Resource States	3-12
3-6 Form for the Resource Entity	3-14
3-7 Form for the Action Entity	3-16
3-8 Initial Form for the Process Entity	3-18
3-9 Form for an Item Passing Process	3-18
3-10 Form for Parameter Passing Process	3-19
3-11 Sample Process Diagram	3-20
3-12 Graphical Representations of Process Primitives	3-23
3-13 Form for an ACTION Primitive	3-24
3-14 Form for the ALLOC Primitive	3-26
3-15 Form for the ASSIGN Primitive	3-28
3-16 Form for the BRANCH Primitive	3-29
3-17 Form for the CALL Primitive	3-31
3-18 Form for the COMMENT Primitive	3-32
3-19 Form for the COMPARE Primitive	3-34
3-20 Form for the CREATE Primitive	3-35
3-21 Form for the DEALLOC Primitive	3-36
3-22 Form for the DESTROY Primitive	3-37
3-23 Form for the ENTRY Primitive	3-38
3-24 Form for the EVAL Primitive	3-42
3-25 Form for the FILE Primitive	3-43
3-26 Form for the FIND Primitive	3-44
3-27 Form for the LOCK Primitive	3-45
3-28 Form for the LOOP Primitive	3-46
3-29 Form for the PROB Primitive	3-47
3-30 Form for the READ Primitive	3-48
3-31 Form for the REMOVE Primitive	3-50
3-32 Form for the RESET Primitive	3-51
3-33 Form for the RESUME Primitive	3-52
3-34 Form for the SEND Primitive	3-53
3-35 Form for the SUSPEND Primitive	3-54
3-36 Form for the TEST Primitive	3-55
3-37 Form for the TRACE Primitive	3-56
3-38 Form for the UNLOCK Primitive	3-58
3-39 Form for the WAIT Primitive	3-59
3-40 Form for the WRITE Primitive	3-60
3-41 Sample Legal Path Table Entries	3-62
3-42 Form for the File Entity	3-64
3-43 Form for the Table Entity	3-67
3-44 Forms for Constant and Variable Entities	3-69
4-1 AISIM Levels of Operation.....	4-2
4-2 Help Database Structure	4-4
4-3 Form for the Guideline Help Topic	4-5
4-4 Form for the Note Help Topic	4-6

FIGURE

PAGE

4-5	Form for the Procedure Help Topic	4-7
5-1	AISIM READY Level Command Summary	5-2
5-2	Sample Batch Job Submission	5-8
5-3	Sample Batch Job Submission with Plots	5-9
6-1	Terminal Profiles	6-2
6-2	Design User Interface Commands	6-4
6-3	DUI Command Summary	6-5
6-4	PEI Command Summary	6-18
6-5	Process Display with Menu	6-29
6-6	Viewspace versus Workspace in ADE	6-37
6-7	ADE Command Summary	6-39
6-8	Architecture Symbols	6-40
6-9	Sample Architecture	6-63
6-10	Sample LPT Generated by Method A	6-63
6-11	Sample LPT Generated by Method B	6-64
6-12	Sample LPT Generated by Method C	6-65
7-1	Form for Text Description	7-1
7-2	Analysis User Interface Commands	7-5
7-3	AUI Command Summary	7-6
7-4	DEFPLOT Form for Items	7-9
7-5	DEFPLOT Forms for Process	7-9
7-6	DEFPLOT Forms for Queues	7-10
7-7	DEFPLOT Forms for Resources	7-10
7-8	DEFPLOT Form for Variables	7-11
7-9	Sample Plot	7-11
7-10	Sample Form for Selecting Plots	7-23
7-11	Sample Plot	7-23
8-1	RUI Command Summary	8-2
10-1	LUI Command Summary	10-2
10-2	Library Utility Data Flow Diagram	10-3
10-3	Checkout Command Summary	10-6
10-4	Mergein Command Summary	10-17
10-5	Mergeout Command Summary	10-26
11-1	HEI Command Summary	11-2
11-2	Update Command Summary	11-8
12-1	File Management User Interface	12-2
12-2	FUI Command Summary	12-3
13-1	Initialization Report - Constants, Files, Tables, and Global Variables	13-3
13-2	Initialization Report - Items and Queues	13-4
13-3	Initialization Report - Resources and Architecture Legal Path Table	13-5
13-4	Initialization Report - Actions and Processes	13-6
13-5	Initialization Report - Loads and Scenario	13-7
13-6	Constant Report	13-8
13-7	Numeric Variable Report	13-9
13-8	Non-numeric Variable Report	13-10
13-9	Item Report	13-11
13-10	Resource Report	13-13
13-11	Action Report	13-15
13-12	Queue Report	13-17
13-13	Process Report	13-20

FIGURE

PAGE

D-1	Listing of Process MRS	D-4
D-2	Listing of Process NODEPROC	D-6
D-3	Listing of Process DESTPROC	D-8
D-4	Listing of Process CHANPROC.....	D-10

SECTION 1

INTRODUCTION

1.1 PURPOSE AND SCOPE

The Automated Interactive Simulation Modeling System (AISIM) provides the user with the ability to do high level simulation of complex operational and distributed data processing systems. The purpose of this manual is to provide the AISIM user with a comprehensive guide for the use of AISIM version 5.0 on a VAX 11/780 computer.

1.2 ORGANIZATION

This manual is organized to serve as a straightforward reference document for the AISIM user. Section 1 introduces this document, detailing the organization of this document, the document conventions and applicable documents. Section 2 is an overview of the concepts used in modeling and simulation of systems using AISIM. Section 3 contains a detailed description of the AISIM modeling constructs. Section 4 describes the interface between the AISIM software and the host computer's time sharing system. Sections 5 through 12 present information of the various system user interface levels, including detailed descriptions of prompts and commands. Section 13 discusses AISIM simulation results and how to interpret them. Appendix A presents operational procedures and other information which is useful for the user to know but not mandatory for using the system. Appendix B lists simulation error messages with a description of their meaning. Appendix C is a glossary of AISIM terms. Appendix D contains a detailed description of the message routing submodel described in section 3.

1.3 DOCUMENTATION CONVENTIONS

The descriptions of AISIM commands given in this manual use the following notations to define the syntax and format of the AISIM commands:

1. Commands shown in the format below are equivalent:

DESIGN

D

The latter is an abbreviation for the former.

2. Required parameters are enclosed in braces:

{entity-type}

3. Optional parameters are enclosed in brackets:

[NOXLATE]

Default values exist for all optional parameters.

4. The brace and bracket symbols are used only to define the format. They should never be typed in the actual command statement.

braces { }

brackets []

5. The symbols listed below should be typed in a command statement exactly as shown in the command statement definition.

apostrophe '

comma ,

parentheses ()

period .

6. Words in lower case appearing in a command definition represent variables for which the user should substitute specific information in the actual command.

EXAMPLE: If "database" appears in a command definition, the user should substitute a specific name of a database (for example, CONTACT) for the variable when the command is entered on the terminal.

7. All upper case words and letters in a command definition, such as a command name or a parameter name, must be typed as part of the command statement.
8. All command names and associated parameters must be separated from each other by the appropriate delimiter, as shown in the command definition. Delimiters are either a comma or a blank depending on the context. A blank is entered on the terminal by pressing the space bar at the bottom of the terminal keyboard.

EXAMPLE: BACKUP [PROJECT(database)]

If the optional parameter is used, it must be separated from the command name BACKUP by a blank (), i.e.,

BACKUP PROJECT(contact)

When a comma is to be used as the delimiter, it will be specified as part of the command definition.

EXAMPLE: DEF PLOT {entity-type},{entity-name}

In this example the command name DEF PLOT would be separated from the required parameter {entity-type} by a blank and the two required parameters would be separated from each other by a comma, i.e.,

DEF PLOT R,resource

9. The references in this document to specific words that are AISIM entities will appear with an initial capital. This is to distinguish the reference to an AISIM specific concept from a common interpretation of the word.

EXAMPLE: Process - Occurrences of this refer to the AISIM entity.

1.4 APPLICABLE DOCUMENTS

The following documents provide additional information relevant to the operation and use of AISIM:

AISIM Training Manual

AISIM Training Examples Manual

SECTION 2

AISIM CONCEPTS

The Automated Interactive Simulation Modeling System (AISIM) provides a tool for the analysis of complex systems. The tool is designed for the operations analyst or engineer as a workbench for investigating the impact of system alternatives. AISIM provides a graphics language for the expression of systems, a database for storing a system's design and a simulation capability for analyzing the system. AISIM is applicable to design analysis of hypothetical systems and to the operations analysis of existing systems.

AISIM is a computer program that allows for the simulation of complex systems by a user without the need for the user to do additional programming. The program can be executed interactively by a user communicating with a host computer through a terminal, or by submitting some AISIM operations to be performed in batch mode. By using the host computer in an interactive mode, an AISIM user can use AISIM to obtain timely data to support decisions on how a system is to function.

2.1 CHARACTERISTICS OF SYSTEMS MODELED BY AISIM

AISIM supports the design and analysis of systems having any of the following characteristics.

1. Procedural operations -- Processes in the system can be described by a sequence of steps that describe the logic of every operation (e.g., operator actions, operating system logic, applications logic, man-machine interface, real time input processing).
2. Parallel Processing -- Any number of processes can occur simultaneously.
3. Shared Resources -- Some processes require resources that are contended for by other processes (e.g., two I/O requests contending for a single channel). Queueing is reflected in the degradation of the time required to complete processes suffering resource contention (e.g., large queues behind bottlenecks in a network).
4. Operational loading -- The operation of the system is a function both of its internal structure and of the environmental pressures on it.
5. Process communication -- Processes transfer data and materials to other processes in the system (e.g., both message routing and network control information communication can be easily represented).

6. Interconnected network -- Network architectures consisting of interconnected nodes can be represented in AISIM. System constructs allow the user to define the routing of messages through the described architecture. AISIM also allows for the modeling of systems abstracted from any particular architecture.

These characteristics are generic to a large class of systems including military, computer, and industrial systems.

2.2 MODELING

In scientific and engineering usage, a model is a simplified (or idealized) representation of a system that is advanced as a basis for calculations, predictions or further investigation. AISIM modeling fits comfortably under this general characterization, but AISIM is especially useful for the modeling of systems which incorporate parallel processing (simultaneous activity) and networks. AISIM is particularly suited to the modeling of embedded computer systems for command, control and communication applications.

There are many applications of simulation modeling in this problem area. AISIM models are representative, discrete event simulation models used for predictive operations analysis. What this means is that entities in a real system are mapped onto AISIM entities that have a very close functional relationship. AISIM entities respond to simulated conditions much like the real entities do under actual conditions. This is in contrast to functional modeling where the real system is described in terms of equations in differential calculus. The emphasis in representative modeling is on describing the system.

Generally, determining and clearly describing the system is the first major obstacle a modeler must confront. If a system is in the design phase, then no data is available on how it will perform or what the major bottlenecks will be. For existing systems these characteristics may be known but the combination of events that cause problems may not be understood. In both cases, much can be learned from modeling the system.

A key concept to keep in mind is that models are a simplified description of a system. This implies that some elements of the real system may not be represented in the model. The challenge in modeling is to represent all the elements of critical interest to the system dynamics in the model. This requires some thought as to the development of the model.

2.3 DESIGNING MODELS

A model should be carefully designed before being built. The key activities addressed during the design phase are the following:

1. Understand the Model and Collect Relevant Data -- To model any system effectively, a modeler has to know something about the system. Building an executable simulation model requires that the system have an accurate and sufficiently detailed description. A modeler must be

aware of the functions performed in a system which affect the dynamics of the operation. A modeler must also know the characteristics of all the elements that perform work, create data, control processing, interrupt normal operations and produce output. This data can be obtained from design specifications, hardware specifications, previous studies or empirical testing. It is important to collect good data because that data becomes the foundation of the model.

2. Determine Model Boundaries -- Systems modeled by AISIM generally consist of many subsystems. The problems caused by the combination of subsystem activities are of interest to the analyst. AISIM provides varying levels of detail in modeling a subsystem. Sometimes the activity can be viewed as a black box. The flow of control through this box can simply be represented by a delay. This type of phenomena is modeled by AISIM with the Action entity. Other times, the characteristics of a subsystem can be represented by a mathematical function. AISIM has such a functional capability with the EVAL Primitive and Table entity. If an activity is more complicated, it can be described by logic. In this case, AISIM allows the modeler to go to his own level of detail by building a Process. Setting the boundaries of an AISIM model is precisely what the modeler does in deciding which of these constructs will be used to model the elements of a system. A method of paper modeling developed for software design is known as "structured design". This method uses structure charts, hierarchical charts showing calling sequences, to describe functional processing. This method has been used successfully with AISIM. An alternate method would be to create flow charts of the various system functions.
3. Determine Experimental Method -- A model allows an analyst to run experiments on a system to predict how an operation will behave. Before any effort is expended in building a model, the desired output of the simulation runs must be considered. Monitors can be designed to provide data on the system's operation. Experiments can be designed to validate the model.

2.4 CONSTRUCTING AN AISIM MODEL

2.4.1 Charting a Paper Model

In building a model, a modeler maps the elements of a system onto the constructs of the simulation language. To do this, the modeler must be familiar with the characteristics and relationships of both the simulation tool and the real-world system. The mapping is not always clear-cut and usually requires iteration. The modeler charts out what processing takes place in a system, where resources are allocated, how processes communicate and where activities initiate. This chart is referred to as a paper model. It may be derived from an understanding of the system's functions and a graphical representation of its network. On the paper model, the modeler names the entities in the system that will be modeled by AISIM entities - Processes, Resources, Items, Queues, Tables, etc.

2.4.2 Defining the AISIM Model

An AISIM model is built by defining AISIM entities to represent system entities. This is done interactively on the computer. AISIM solicits relevant data for defining all design entities.

2.5 AISIM MODELING ENTITIES

As mentioned earlier, a model is a description or abstraction of a real or proposed system. To build a model with the intention of simulating its operation, we must describe the model in terms which can be interpreted, and operated upon, by the simulation system. That is, a system can be modeled using a prose description; but unless it has some systematic relation to a computer language, it would be useless as a computer model because prose is ambiguous. AISIM uses a special set of terms to describe system structure and operation called AISIM entities. A modeler must understand the meaning and use of these entities to build successful models. These entities are briefly discussed below. A detailed discussion of each of these entities is provided in section 3. Figure 2-1 also provides further insights to the meaning, use, and relationships between entities and other modeling constructs.

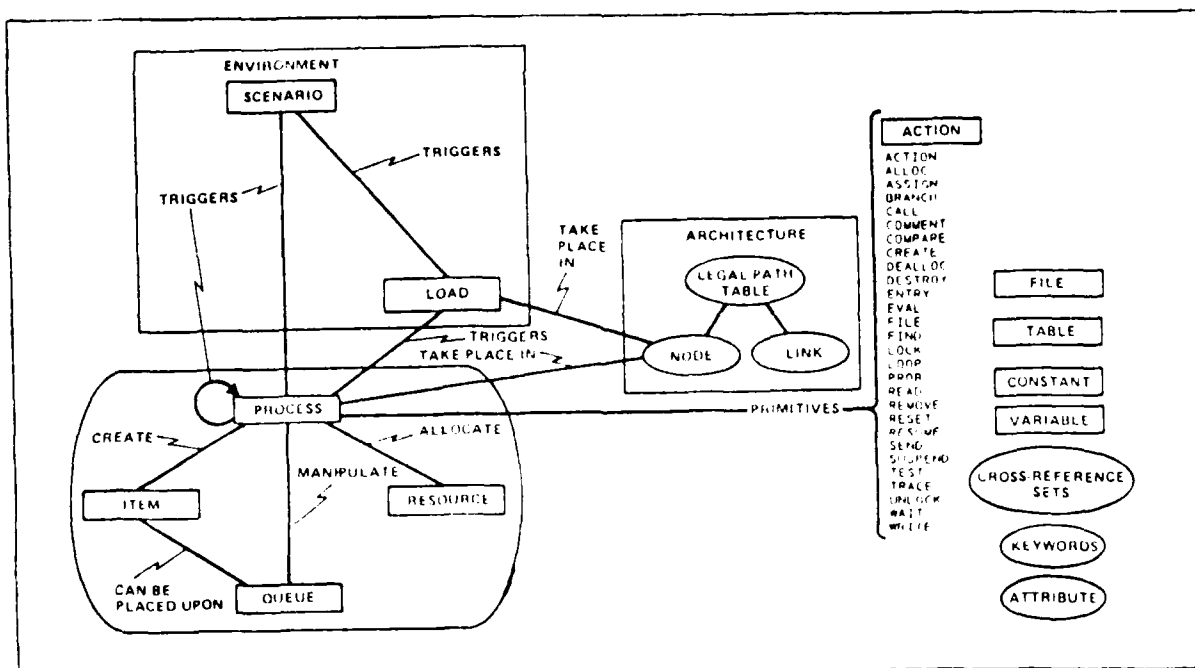


Figure 2-1. AISIM Entity Relationships

Action - An Action entity is used to represent the consumption of time for any action, activity, decision, etc., that consumes time. Each Action entity corresponds to an ACTION Primitive in a Process. The ACTION Primitive is the only one that causes the simulation clock to be updated.

Constant - A Constant is a model parameter whose value does not change during a simulation exercise of a model. Constants are used to represent parameters that do not vary with time or in response to the workings of the system being modeled.

File - A File entity corresponds to an external file from which data is read or to which data is written during a simulation run.

Item - An Item is a transient data element and is used to represent messages (or materials or even physical objects) flowing through the system.

Load - A Load is used to represent aspects of the world outside the system that trigger the initiation of Processes. Loads represent the normal burden, i.e., occasional Process triggering, on a system.

Primitive - Primitives are logical constructs that represent steps in the modeled system's operation. There are 28 different Primitives each representing a different logical function. A sequence of Primitives compose a Process. All of the Primitives are listed below.

ACTION
ALLOC
ASSIGN
BRANCH
CALL
COMMENT
COMPARE
CREATE
DEALLOC
DESTROY
ENTRY
EVAL
FILE
FIND
LOCK
LOOP
PROB
READ
REMOVE
RESET
RESUME

SEND
SUSPEND
TEST
TRACE
UNLOCK
WAIT
WRITE

Process - A Process is a logical description (using Primitives) of some or all of the operations, decisions or activities of the system being modeled.

Queue - The Queue entity is used to model an ordered holding area for one or more Items. A Queue may be used to model, for example, a job queue or a memory buffer. A Queue may be defined with a maximum size parameter to model such limits as the maximum number of messages that a buffer can hold before it is overloaded. Queues bear a default size of infinite.

Resource - The Resource entity is used to model the mechanisms (people, CPU, communication lines, etc.) necessary to complete a Process. Resources generally have the property of being shared among Processes. Performance of a Process can be degraded due to contention for Resources.

Scenario - The Scenario entity is used to model the various environments in which a system must perform. A Scenario specifies the number of periods of a simulation run as well as their length (which is uniform). The Scenario schedules the initiation of Loads. It can also schedule the initiation of Processes.

Table - A Table is a user-definable function with up to fifteen pairs of data points. Tables may be defined as either continuous, discrete or alpha. A continuous Table interpolates linearly between numeric points. A discrete Table is a step function connecting numeric points. Alpha Tables are used for structuring data over non-numeric ranges and domains.

Variable - A Variable is a model parameter whose value can change during a simulation run, either by setting it equal to a mathematical expression or through reassignment by the user between stages of a simulation.

Keywords - The keywords are system-defined variables that provide the user with information about the current state of the simulation.

Alpha Literals - Alpha Literals are character strings that are used to make models more readable and are used in comparison with each other to determine process execution control.

SECTION 3

AISIM ENTITIES AND OTHER MODELING CONSTRUCTS

In this section AISIM's entities and other modeling constructs are described in detail. For each entity, the parameters required to define the entity and the means by which this data is requested from the user are described. Included is mention of relationships between the various AISIM entities, where such mention is deemed helpful.

Note: Whenever entity data is requested via a "form" such as that for the Scenario entity shown in figure 3-1, any information which is a default will appear in the form when the form is presented to the user. In the figures contained in this manual the defaults appear as white words in the black form.

SCENARIO

3.1 SCENARIO

The Scenario entity is used to represent the various environments in which the system being modeled must perform. Together with the Load entity it represents the external stimuli on a modeled system.

In a Scenario, the user defines a collection of Loads and/or Processes, together with schedule time and triggering priority for each. The Scenario calls for the initiation of activity over time by activating a Process or Load at the corresponding scheduled time.

Scenarios are divided into periods whose length and number are chosen by the user. These periods provide break points at which the user can stop a simulation to alter a variable or inspect the results up to that point. There may be up to 14 periods in a given Scenario. The form for the Scenario is shown in figure 3-1.

SCENARIO:

PERIOD LENGTH: UNITS: OUTPUT UNITS:

DESCRIPTION:

PERIODS:

CALLS:

TRIGGER	SCH TIME	UNITS	PRIORITY	TRIGGER	SCH TIME	UNITS	PRIORITY
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	
		SECONDS				SECONDS	

Figure 3-1. Form for the Scenario Entity

Following is a description of the fields in the Scenario form:

- SCENARIO: Name of the Scenario (1 to 8 characters)
- PERIOD LENGTH: Amount of time in each simulated period.
- UNITS: The time units in which the PERIOD LENGTH is expressed.
- OUTPUT UNITS: These are the time units in which the simulation will be run, (i.e. all time unit specifications throughout the model will be converted to this unit) and in which all simulation output will appear unless changed by the user during the simulation run (see section 7.15).

DESCRIPTION: Any user comment (0 to 53 characters)

PERIODS: Any characters may be entered in these fields. The number of fields containing characters determines the number of periods in a simulation, i.e., for each of the 14 fields in which an entry is made a period is added to the total simulation run. A Scenario can have a maximum of 14 periods.

TRIGGER: The name of a Process or Load that is to be initiated at the scheduled time.

SCH TIME: The simulation time, from the start of the simulation, at which the the Load or Process specified is to be initiated.

UNITS: The time units in which SCH TIME is expressed.

The valid entries for UNITS, OUTPUT UNITS and schedule UNITS are as follows:

<u>Form Entry</u>		<u>Meaning</u>
nseconds (ns)	-	nanoseconds
useconds (us)	-	microseconds
mseconds (ms)	-	milliseconds
seconds (s)	-	seconds
minutes (m)	-	minutes
hours (h)	-	hours
days (d)	-	days

The default value which is automatically placed in the form is SECONDS, but the user can change this default by using the Design User Interface UNITS command (see section 6.1.9).

PRIORITY: The priority the triggered Process is to have. Priority is inverse, priority 1 preempts priority 2. If a Load name is entered in the trigger field, the corresponding priority field is ignored.

Operation - A model database may contain more than one Scenario. However, only one Scenario can be used per simulation run. The Scenario specified will define the simulation period length, and Loads and Processes to be triggered by the Scenario. The total simulation time is the product of the number of periods and the period length. The number of periods also effects the collection of plot data points. (see appendix A.3)

Constants may be used to define PERIOD LENGTH, SCH TIME, and PRIORITY in order to parameterize a model.

Scenario entities are entered into a model by using the Design User Interface EDIT command (see section 6.1.4).

3.2 LOAD

The Load entity is used with the Scenario entity to periodically trigger Processes during the simulation, and optionally at specific nodes in the architecture. The Load describes which Processes will be initiated and at which nodes. An instance of the Load is triggered simultaneously at each of the specified nodes. This entity can be described in the following way: for each Process in the Load, initiate up to the maximum number at an interval determined by the schedule method, and initiate them simultaneously at each of the specified nodes. The form for the LOAD entity is shown in figure 3-2.

LOAD:

NODE1	NODE2	NODE3	NODE4
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
NODE5	NODE6	NODE7	NODE8
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

DESCRIPTION:

PROCESS	MAX #	SCHMTD	MEAN	DELTA	UNITS	PRIORITY
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	SECONDS	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	SECONDS	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	SECONDS	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	SECONDS	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	SECONDS	<input type="text"/>

Figure 3-2. Form for the Load Entity

Following is a description of the fields in the Load form.

- LOAD: Name of the Load (1 to 8 characters)
- NODES: If an architecture is used these are the nodes in which the Processes specified will take place. Otherwise leave blank.
- DESCR: Any user comment (0 to 53 characters)
- PROCESS: Names of Processes which the Load triggers according to schedule.
- MAX #: Maximum number of times this Process is to be triggered in each execution of the Load.
- SCHMTD: Statistical function to be used to determine the time between Process triggerings. It can be any of those described under SCHEDULE METHODS (see below).

MEAN: Depending upon schedule method, MEAN is used to determine the interval between each triggering of a Process. In general this is the mean inter-arrival time.

DELTA: Depending upon schedule method, DELTA is used to determine the deviation about the mean for the interval between triggerings of a Process.

UNITS: The time units in which the schedule is expressed. The valid entries are as follows:

<u>Form Entry</u>		<u>Meaning</u>
nseconds (ns)	-	nanoseconds
useconds (us)	-	microseconds
mseconds (ms)	-	milliseconds
seconds (s)	-	seconds
minutes (m)	-	minutes
hours (h)	-	hours
days (d)	-	days

The default value which is automatically placed in the form is SECONDS, but the user can change this default by using the Design User Interface UNITS command (see section 6.1.9).

PRIORITY: Priority with which the Process is to be executed. Priority is inverse, priority 1 preempts priority 2. Priority is used to determine which Process will be allowed to allocate a Resource when it is contended for by two or more Processes (see ALLOC Primitive, section 3.9.2).

SCHEDULE METHODS:

START - MEAN: inapplicable; i.e., leave field blank
 DELTA: inapplicable; i.e., leave field blank

All Processes up to the maximum number are initiated at the same clock time, the start of the Load. This can be used to simulate pre-loading.

INTERVAL - MEAN: time between initiations
 DELTA: inapplicable; i.e., leave field blank

One Process is initiated at every interval as defined by MEAN. The first starts at the time given by MEAN with respect to the starting time of the Load.

POISSON - MAX #: mean number in a PERIOD
 MEAN: inapplicable; i.e., leave field blank
 DELTA: inapplicable; i.e., leave field blank

Processes are scheduled randomly by a Poisson process. The time between Process triggerings is exponentially distributed. The MAX # parameter defines the mean number for a PERIOD. PERIOD length is defined in the Scenario.

EXPONENT - MEAN: mean time between Process triggerings
DELTA: inapplicable; i.e., leave field blank

The time passing between Process triggerings is exponentially distributed.

LOGNORML - MEAN: mean time between Process triggerings
DELTA: standard deviation of time between Process triggerings

The time passing between Process triggerings is lognormally distributed.

NORMAL - MEAN: mean time between Process triggerings
DELTA: standard deviation of time between Process triggerings

The time passing between Process triggerings is normally distributed.

UNIFORM - MEAN: mean time between Process triggerings
DELTA: range about the MEAN

The time passing between Process triggerings is uniformly distributed. The DELTA parameter specifies the difference between the largest possible time between Process triggerings and the MEAN time.

ERLANG - MEAN: mean time between Process triggerings
DELTA: order of the distribution function

The time passing between Process triggerings is Erlang distributed. The order "k" is given by the DELTA.

WEIBULL - MEAN: scale parameter.
DELTA: shape parameter

The time passing between Process triggerings is Weibull distributed.

GAMMA - MEAN: mean time between Process triggerings
DELTA: k

The time passing between Process triggerings is gamma distributed.

Operation - a Load specifies a cluster of Processes to be triggered according to a scheduling method and a priority.

Relationships - Loads are part of Scenarios and specify Processes to be triggered and nodes in which they are to be triggered.

Load entities are entered into a model by using the Design User Interface EDIT command (see section 6.1.4).

3.3 ITEM

The Item entity is used to model transient data elements that "flow" through a system. These data items, which, by the nature of their varying attribute values, permit data dependent decision making and timing.

Items can be originated, terminated and passed through the system from one Process to another via the Primitives CREATE, DESTROY, CALL and SEND. Items can also be placed on and removed from Queues via the Primitives FILE and REMOVE, and pointed to via the Primitive FIND. The form for the Item entity is shown in figure 3-3.

ITEM NAME:

DESCRIPTION:

ATTRIBUTES

NAME	VALUE	NAME	VALUE
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Figure 3-3. Form for the Item Entity

Following is a description of the fields in the Item form.

ITEM NAME: Name of the Item (1 to 8 character)

DESCRIPTION: Any user comment (0 to 53 characters)

NAME: Name of an attribute of the Item. An Item can have up to 15 user-defined attributes.

VALUE: The initial value to be assigned to the corresponding attribute (integer, decimal, or character); if character, it must be a defined Process, Resource, global Variable, Constant, Item, Queue, Table or Action or a keyword or alpha literal.

NOTE: All Items have two implicitly defined attributes, TAIL and PRIORITY. TAIL is the number of the Item created, and PRIORITY is the priority of the Process that creates the Item. The TAIL attribute can be used for Item matching (see SEND Primitive).

Operation - An Item is created for each occurrence of the following:

- a. a CREATE Primitive that is executed - used to model transient data elements
- b. a SEND Primitive that is executed in a Process which does not have an Item of the specific name attached at the time.

An Item is terminated only when the DESTROY Primitive is executed.

Attribute values are assigned at the time of creation.

Relationship- Item attributes are used by Process Primitives and attribute values can be modified by the ASSIGN and READ Primitives.

Item entities are entered into a model by using the Design User Interface EDIT command (see section 6.1.4).

3.4 USER DEFINED QUEUES

A Queue is a global entity used to represent an ordered holding area for Items.

When a Queue is defined, a maximum size parameter is specified (the default is "infinite"). This allows Queues to model finite storage devices that have a limited capacity (e.g., a storage bin, a computer job scheduler). Once the value is defined, it may not be changed and thus this parameter must be either a numeric value or a defined Constant.

Queues are manipulated by Processes through the use of the FILE, FIND, and REMOVE Primitives. An Item may be placed on a Queue, if space exists, by using the FILE Primitive, specifying one of four location parameters: FIRST, LAST, BEFORE and NEXT. The former two parameters denote the end points of a Queue, the first and last slots. The latter two are location parameters relative to a Queue pointer (see below). If no space exists on the Queue, the Process which is executing the FILE Primitive is suspended. This condition is known as Queue blocked. In this state the Process waits until space becomes available on the Queue. Waiting for space on a Queue is by a first come first served discipline.

An Item may be taken off the Queue by using the REMOVE Primitive and specifying a location parameter (i.e., FIRST, LAST, or NEXT, where NEXT means the current Item pointed to by the Queue pointer). After an Item is removed from a Queue, it may be sent, destroyed, or otherwise modified.

An Item may not be modified, sent, or destroyed while it is on a Queue. The same Item instance may not exist on more than one Queue. Multiple Processes may access the same Queue.

A Queue pointer is maintained for each Process which references a Queue. This pointer contains the address of the entity that the Process is addressing in a Queue. The contents of the Queue pointer is determined by rules described below and in the sections on the Primitives FILE (section 3.9.13), FIND (section 3.9.14) and REMOVE (section 3.9.19):

1. The pointer contains the address of the last entity found with a FIND Primitive; otherwise,
2. The pointer contains the address of the last entity filed with a FILE Primitive; otherwise,
3. The pointer contains the address of the successor of the last entity removed with a REMOVE Primitive with a NEXT option.

The REMOVE and FIND Primitives access a Queue and set the value of the local variable referenced in the Primitive. This means that when a FIND or REMOVE Primitive is executed, the value of the local variable could be set to 0. This occurs under the following circumstances:

1. A REMOVE Primitive attempts to remove an entity from an empty Queue.
2. A FIND Primitive accesses an empty Queue.
3. The NEXT or BEFORE Item in a Queue does not exist.

The form for the Queue entity is shown in figure 3-4.

QUEUE: SIZE: INFINITE
DESCR:

Figure 3-4. Form for the Queue Entity

Following is a description of the fields in the Queue form.

QUEUE: Name of the Queue (1 to 8 character)

SIZE: An integer value of 1 to 8 digits, a defined Constant entity, or the word INFINITE (which is the default)

DESCR: Any user comment (0 to 53 characters)

Relationships - Queues are used to hold Items. Queues are manipulated by the FILE, FIND, and REMOVE Primitives.

Queue entities are entered into a model by using the Design User Interface EDIT command (see section 6.1.4). Attributes associated with Queue entities are described in section 3.13.

See section 3.5 for a description of system defined queues.

3.5 SYSTEM DEFINED QUEUES

System defined queues are managed by the simulator during a simulation run. Queues are used to manage resources and the names of AISIM entities.

3.5.1 States Associated with Resources

Associated with each Resource entity are five simulation states upon which statistics are kept. Four of these states apply to Resource units and one of the states applies to Processes. Resource units can be in one of the four states idle, busy, hold, and inactive. If a Process is waiting for a Resource unit which is unavailable, the Process is in the wait state. Resource units which are idle or inactive are accounted for by counters associated with the Resource. Busy Resource units are kept on a system-defined queue called the busy queue. Resource units that are part of a multiple-unit request are held in a system-defined hold queue until all requested units are available. Processes which are waiting for Resource units are kept on a wait queue. Resources and Processes are placed in these states during the simulation as follows:

Resource units are idle while they are unallocated and available to Processes. Resource units are in the idle state: (1) at the initialization of the simulation, (2) when removed from the inactive state (by the RESET Primitive) or (3) when removed from the busy queue (by the DEALLOC Primitive).

Resource units are placed on the busy queue while they are allocated by some Process through the ALLOC Primitive. They may be removed from the busy queue (1) by being deallocated with the DEALLOC Primitive or (2) by being set inactive by the RESET Primitive.

Resource units are in the inactive state when they are not available to be allocated by Processes. Resources may be placed in this state (1) at the initialization of the simulation, (2) from the idle state by means of the RESET Primitive, and (3) from the busy state by means of the RESET Primitive.

When some of the Resource Units of a multiple-unit request are available before the complete request can be filled, the available units are placed in a hold queue until the remaining units become available. When all of the Resource units requested become available, they are placed on the busy queue as described above.

The wait queue holds Processes that are suspended for lack of an available unit of the needed Resource. A Process is placed on this queue when either (1) it attempts to allocate the Resource (with the ALLOC Primitive) that is held by another Process of equal or "higher" priority or (2) it loses a Resource to a "higher" priority Process.

The relation between these states is illustrated in figure 3-5.

During a simulation run statistics are kept on the activity of these states. These results are presented in the simulation's Resource report. The user can access the number of Resource units or Processes currently in the idle, busy, inactive, or wait states using attributes described in section 3.13.

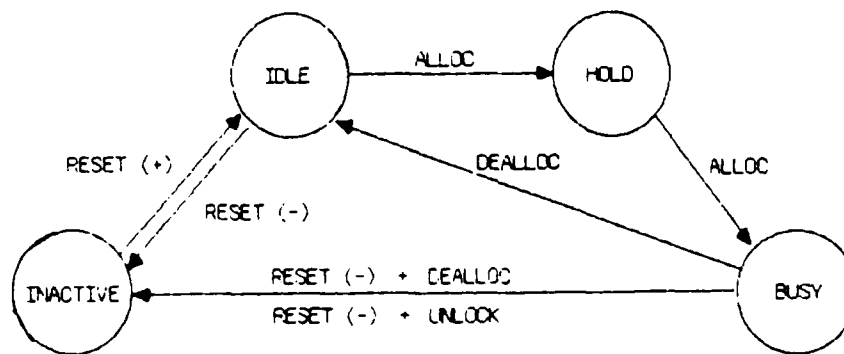


Figure 3-5. Resource States

3.5.2 Cross Reference Sets

In addition to the queues associated with Resource contention, there are eight system defined queues called "cross-reference sets". These queues correspond to the sets of names of the following AISIM entities:

1. Resource names
2. Queue names
3. Process names
4. Item names
5. Action names
6. Table names
7. Constant names
8. Variable names

What this means is that an AISIM modeler can write Processes which perform some function on each entity defined in one of the above sets.

The FIND Primitive accesses the set of names of an entity type by specifying the name, e.g., Resource, Item, Process, as the Queue field reference in the Primitive.

3.6 RESOURCE

The Resource entity is used to model the mechanisms required to perform a Process. "Mechanisms" in this context can be computer processors, memory, communications channels, support personnel, documents, etc.

Queueing for a Resource occurs only within a Process and, in particular, only where an ALLOC Primitive is used. In other words, if no ALLOC Primitive is used there will be no queueing. If no Resource is used (allocated) within a Process, the Process can be executed in parallel (simultaneously) by any number of concurrent requests and the model will represent only time delays associated with the ACTION Primitive.

When a Resource is used (allocated) by a Process, there can be only as many concurrent executions of the Process as there are Resource units available. For example, if the capacity of a Resource is one, then any Processes which allocate that Resource will be executed serially (one at a time). Execution concurrency is controlled only between the allocation and deallocation of the Resource (i.e., if the ALLOC Primitive is the second Primitive in a Process, the first Primitive can be executed concurrently by any number of requests whereas the ALLOC Primitive can be executed concurrently by only as many requests as the Resource has units available).

If no Resource units are available (i.e., idle or presently allocated to a lower priority Process) when an ALLOC Primitive is attempted, the Process' allocation request is merged onto a wait queue associated with the Resource. How the request is merged depends on the priority of the request. The request is merged and sorted by inverse priority (priority 1 preempts priority 2). Within priority the sorting is done first-in-first-out. When deallocation of the Resource (by some other Process) has resulted in enough units to satisfy the requests, and the request has moved to the top of the wait queue, then the request is removed from the queue, the allocation is performed, and the Process is executed. Note that a deallocation of several units may result in several requests being removed from the queue simultaneously. For allocation requests of multiple units, the user can specify whether the units are to be allocated as they become available or only when all units are available at the same time.

If, when the ALLOC Primitive is attempted, there is a lower priority Process possessing the desired Resource units, then the higher priority Process will "steal" those units. The lower priority Process will be suspended while it waits for Resource units. It will be placed on the wait queue but its seniority is based upon the time of its first allocation attempt, not the time it lost its Resources. If at the time of a multiple-unit request some of the units are available, and the request is for units to be allocated as they become available, the currently available units are placed in a hold queue until the request is filled. As units become available, they are added to the hold queue until all of the requested units are available. Then the units are moved to the busy queue.

The Resource entity provides the most interesting and useful simulation results; e.g., delays, bottlenecks, utilization percentages, and traffic statistics. Therefore, the use of Resources should be carefully designed from both the standpoint of model credibility and the specification of required simulation output.

The form for the Resource entity is shown in figure 3-6.

RESOURCE NAME:

TOTAL NUMBER OF UNITS:

INITIAL NUMBER OF UNITS:

DESCRIPTION:

ATTRIBUTES

NAME	VALUE	NAME	VALUE
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Figure 3-6. Form for the Resource Entity

Following is a description of the fields in the Resource form:

- RESOURCE NAME: 1 to 8 character name of Resource
- TOTAL NUMBER: Maximum number of units of the Resource that can be allocated (integer or named Constant).
- INITIAL NUMBER: Number of units available for allocation at the start of the simulation (integer or named Constant).
- DESCRIPTION: Any user comment (0 to 53 characters)
- NAME: 1 to 8 character name of user defined attribute
- VALUE: Initial value to be assigned to an attribute; can be single precision real or integer number, or the name of a defined Variable, Constant, Process, Item, Resource, Queue, Action, Table, or a keyword or alpha literal.

Operation - Resources are initialized at beginning of simulation to the values given above.

Relationships - Resources are used by Processes with the ALLOC, DEALLOC, RESET, LOCK, UNLOCK and TEST Primitives.

Resource entities are entered into the model by using the Design User Interface EDIT command (see section 6.1.4).

In the form for this definition, the Action field contains a name identical with one that appears in an ACTION Primitive in a Process. DESCRIPTION is used to describe the Action. When AISIM automatically creates an Action entity, the description is copied from the COMMENT field of the Action Primitive (see section 3.9.1), but the user can change the description by using the Design User Interface EDIT command (see section 6.1.4). The form for the Action entity is shown in figure 3-7.

ACTION: [REDACTED]

DESCRIPTION: [REDACTED]

Following is a description of the fields in the Action form:

ACTION: 1 to 8 character name of action

DESCRIPTION: Any user comment (0 to 53 characters)

Note: Although AISIM automatically creates and deletes all Action entities as a user places and deletes ACTION Primitives, the user is allowed to create and delete Action entities. For example, a user would need to create an Action entity in the case where the name of an ACTION is passed to a Process and the name used in the ACTION Primitive in that Process is only a local variable which will take on the name of the ACTION passed into it.

3.8 PROCESS

The Process entity is used to represent the sequential logic and activities, operations, functions, etc., of the modeled system. Processes are composed of Primitives, each of which represents a step in the function being modeled by the Process. It is at the Primitive level that Resources are allocated and deallocated, time is consumed, decisions take place, etc.

In the graphic representation of a Process, the Primitives are flanked at the top and bottom by figures labeled START and END. These figures represent the logical entry and exit points for the Process.

Processes are initiated by (1) Scenarios and Loads (within Scenarios) and (2) by other Processes through the CALL and SEND Primitives. Once initiated, the execution of the Process depends upon the availability of the Resources that the Process references through the ALLOC and DEALLOC Primitives.

There are three types of Processes: parameter passing, Item passing, and standard. Each differs in how it is triggered.

A parameter passing Process is one that takes values of local variables from another Process as inputs and/or returns the values of local variables to the other Process as outputs. Such Processes can be triggered only by a CALL Primitive and it is the calling Process which sets up the relation for the values given and returned (see CALL Primitive, section 3.9.5). The given and return values can be numerics, string literals, keywords, the names of Items, Queues, Resources, Processes, Tables and Actions.

An Item passing Process is one that is triggered by having Item(s) delivered to it from other Process(es) through the SEND Primitive. The required Items need not be delivered from a single Process; the sending Processes may be as many as six, but the Process will not execute until all of the Items indicated in the definition are delivered.

A standard Process is one which neither requires Items nor is given (or returns) parameters. It may be triggered by either a CALL Primitive from another Process or through the Scenario or Loads.

When a Process is defined, the node in which the Process is to execute is specified. If the Process can execute in any node, or if there is no architecture, ALL can be specified. Generally, when a Process is triggered, it executes in the same node as its parent, or when a Process is triggered from a Load, the Load nodes specify where the Process is to execute. However, if a Process is triggered from a Scenario, the node specified for the Process is the one in which the Process executes. The node specified in the Process definition is also available to the user through the \$NODE keyword (see section 3.17).

The initial form for the Process entity is shown in figure 3-8.

START
PROCESS NAME [REDACTED] NODE [REDACTED]
ATTRIBUTES ATTACHED (YES OR NO) [REDACTED]
PROCESS DESCRIPTION
[REDACTED]
START BLOCK TYPE [REDACTED]
ENTER "PARM" FOR PARAMETER PASSING
ENTER "ITEM" FOR ITEM PASSING
ENTER "STD " FOR STANDARD PROCESS

Figure 3-8. Initial Form for the Process Entity

Following is a description of the fields in the initial Process entity form:

PROCESS NAME: 1 to 8 character name of Process
NODE: architecture node in which this Process is to execute (if its execution is restricted to a specific node; ALL in this field indicates the Process may execute in any node)
ATTRIBUTES ATTACHED: YES or NO to indicate whether the Process has attributes.
PROCESS DESCRIPTION: 0 to 53 alphanumeric character description.
START BLOCK TYPE: (STD, ITEM, PARM)

To define an Item passing Process the user enters "Item" in the START BLOCK TYPE field. The user will then be presented with the form shown in figure 3-9.

ITEM PASSING START
ITEMS RECEIVED:
[REDACTED] [REDACTED] [REDACTED]
MUST ALL THE ITEM SERIAL NUMBERS MATCH (Y/N) [REDACTED]

Figure 3-9. Form for an Item Passing Process

This form is for providing a list of the needed Items. The Items received by each must be of distinct types.

The field concerning the matching of serial numbers asks whether the TAIL numbers (which is a default attribute of every Item) must be the same for all the Items in the Process. If the user enters "Yes" in this field, the Process will not execute until it has received Items of the specified type to which the same TAIL number attribute has been assigned.

To define a parameter passing Process the user enters "PARM" in the START BLOCK TYPE field. The user will then be presented with the form shown in figure 3-10.

PARAMETERS FOR PASSING START

GIVEN:

--	--	--

RETURN:

--	--	--

Figure 3-10. Form for Parameter Passing Process

This form is for providing the names of the local variables to be given and returned to any Process that calls it through the CALL Primitive. The CALL Primitive must contain the same number of entries in its given and return lists as the called Process. If the CALL Primitive does not give or return all the necessary values, an execution error will occur indicating a disagreement in the number of values.

To define a standard Process the user enters "STD" in the START BLOCK TYPE field. Since no inputs are relevant to its execution, there is no secondary form for the definition of a standard Process.

Figure 3-11 is a typical flowchart representation of a Process. This graphical representation of the logic of a Process is presented to the user during the design of an AISIM model.

Relationships - Processes are constructed from Primitives. Resources are used by the Process through the ALLOC, DEALLOC, RESET, LOCK, UNLOCK, and TEST Primitives. Time is consumed by the ACTION Primitive. Processes are initiated by Loads, Scenarios and by other Processes through the CALL and SEND Primitives.

Process entities are entered into a model by using the Design User Interface EDIT command (see section 5.1.4).

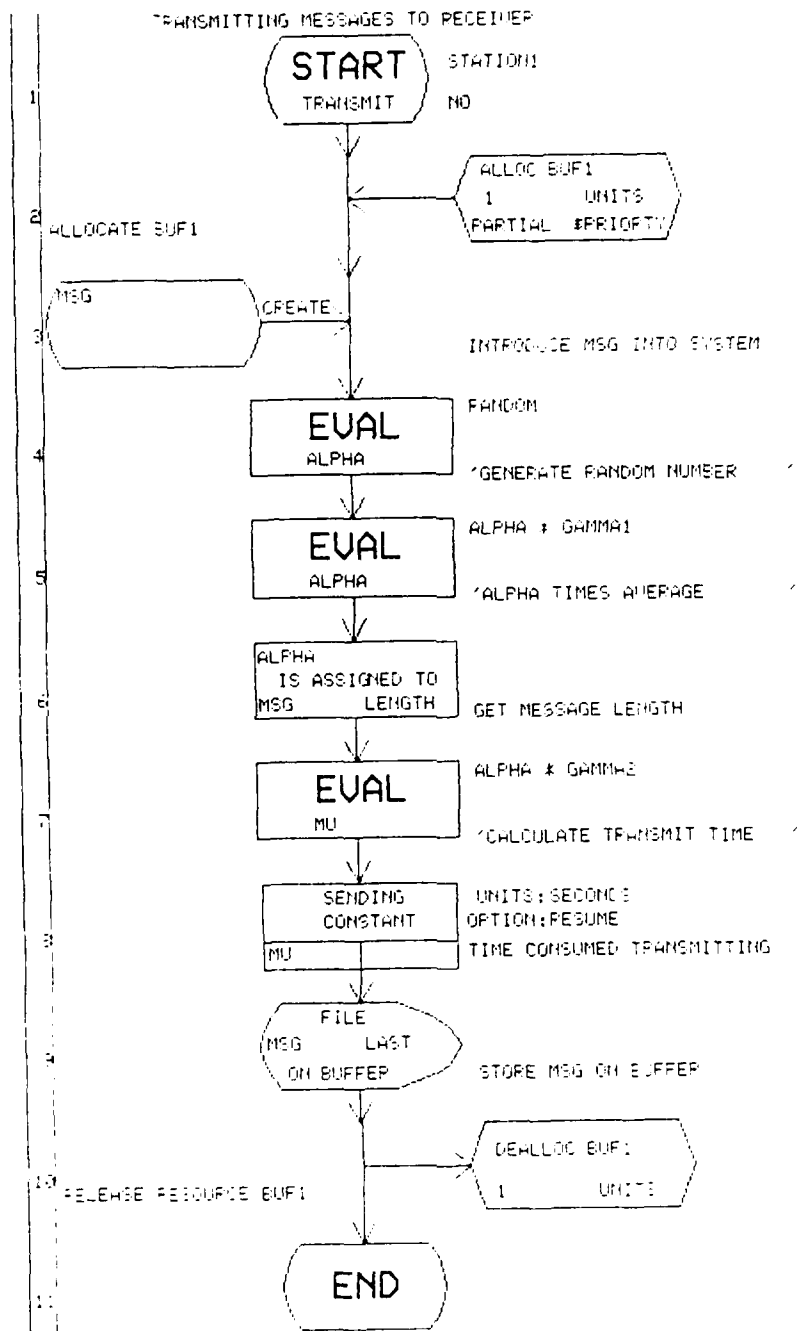


Figure 3-11. Sample Process Diagram

PRIMITIVES

3.9 PRIMITIVES

Primitives are the constituent elements of Processes and are used to characterize procedural steps by sequential logic. AISIM offers a list of 19 Primitives. Although limited in number, the Primitives have been shown to represent all logical operations for computer system modeling. The Primitives can be grouped into eleven functional categories. These categories are as follows:

Process Execution Control

CALL
SEND
SUSPEND
RESUME
WAIT

These Primitives control the initiation and sequencing of Processes.

Branch Control

COMPARE
BRANCH
ENTRY
PROB
LOOP

These Primitives govern the internal branching in the logic of a Process.

Item Handling

CREATE
DESTROY

These two Primitives govern the introduction and elimination of a system's transient data elements.

Time Consumption

ACTION

This Primitive represents the consumption of time through some activity, decision, etc.

Mathematical Operations

EVAL

This Primitive governs calculations, invoking standard mathematical functions and operations or making use of user-defined Tables.

Resource Allocation

ALLOC
DEALLOC
RESET
TEST
LOCK
UNLOCK

These Primitives govern the use of Resources.

Queue Manipulation

FILE
FIND
REMOVE

These Primitives govern storage and retrieval on Queues.

Variable Assignment

ASSIGN

This Primitive governs the assignment of values to Variables or Attributes (both numerical and non-numerical).

Debugging

TRACE

This Primitive has the special function of creating a record of the sequence of Process Primitive executions which takes place during simulation. It is used for debugging and validating a model.

Input/Output

READ
WRITE

These Primitives enable a user to read data from and write data to external files during a simulation run. This data can be used to control the execution of the simulation and to provide data for debugging and validating a model.

Description

COMMENT

This Primitive has no functionality during a simulation run, and is used simply to comment the surrounding logic in a Process.

Figure 3-12 shows the graphic representation of each Primitive, and following is a description of the meaning of each Primitive and the parameters necessary to define each. Primitives are entered using the Process Editor Interface of the Design User Interface (see section 6.2).

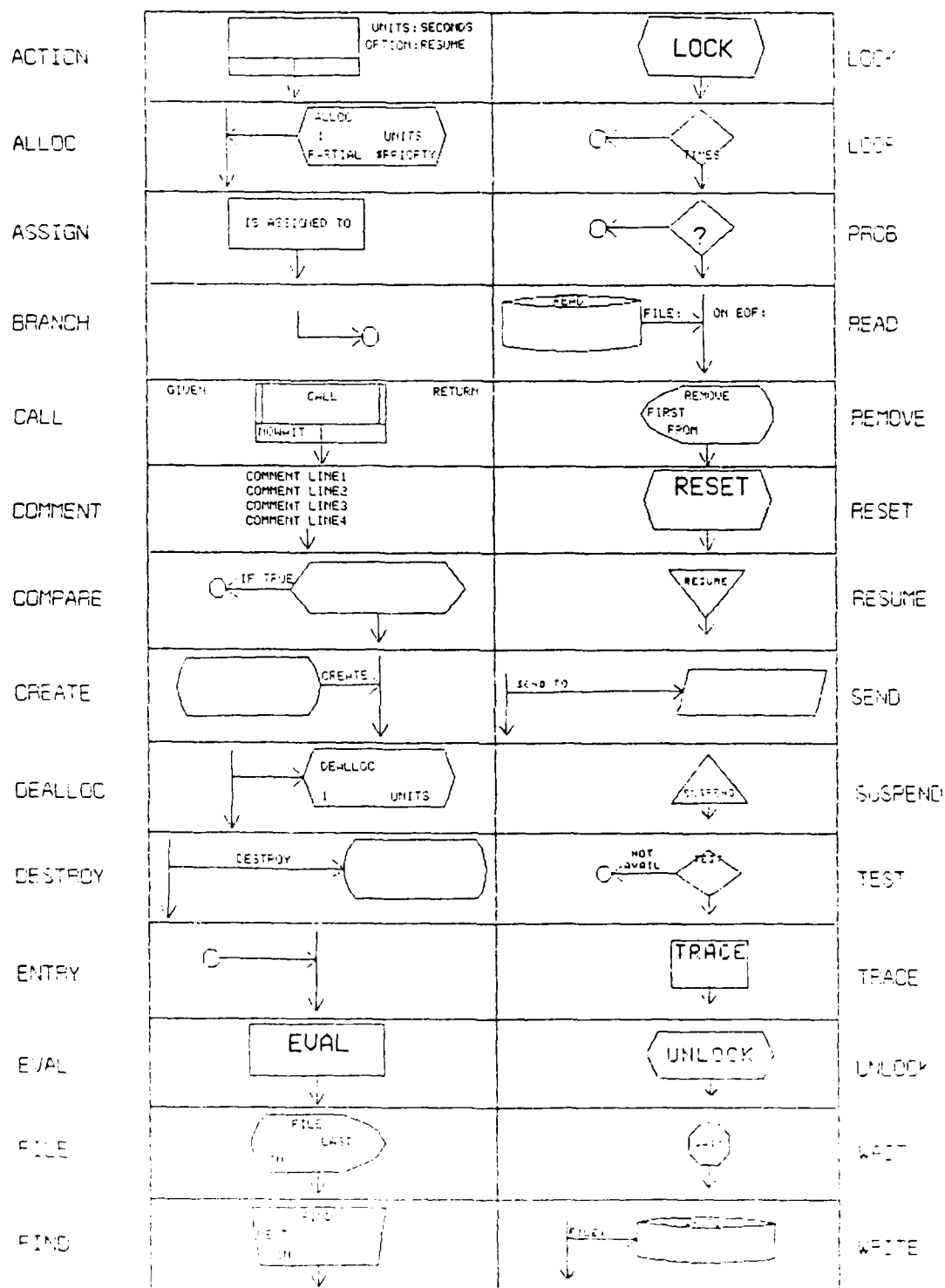


Figure 3-12. Graphical Representations of Process Primitives

3.9.1 ACTION

The ACTION Primitive represents the consumption of time for an activity that consumes time. The ACTION Primitive is used to model the time to perform some real work event such as a man's activity or a machine's activity. An Action Primitive can represent both interruptible and noninterruptible tasks (i.e., tasks which can start up where they left off after being stopped due to a loss of Resources, or tasks which must be completed in one uninterrupted session). The time consumed by an ACTION Primitive is determined according to the selected distribution function (described below). The form for an ACTION Primitive is shown in figure 3-13.

PARAMETERS FOR ACTION

ACTION NAME: OPTION: RESUME METHOD:
 MEAN TIME: DELTA TIME: UNITS: SECONDS
 COMMENT:

Figure 3-13. Form for an ACTION Primitive

Following is a description of the fields of an ACTION form:

ACTION NAME: A name assigned to the Action.

OPTION: Specifies disposition of the Action upon regaining Resources that were lost due to pre-emption by a higher priority Process. Valid options are RESTART and RESUME. RESTART indicates that the Action must be restarted after being interrupted. RESUME (the default) indicates that the Action can continue where it left off.

METHOD: Distribution function type, which may be: CONSTANT, EXPONENT, LOGNORML, NORMAL, UNIFORM, GAMMA, ERLANG or WEIBULL. (The random number seed used for statistical functions can be controlled by the user in the AUI.)

MEAN TIME: Typically specifies the average duration time of the Action. This parameter varies in meaning depending on the METHOD selected. For CONSTANT, it specifies the exact duration value. For WEIBULL, it specifies the distribution's scale parameter. For all other methods, it specifies the mean duration.

DELTA TIME: This parameter varies in meaning depending on the METHOD selected. Typically it specifies the variation, about the mean, in the duration times. Specifically:

CONSTANT - inapplicable (i.e., leave field blank)
EXPONENT - inapplicable (i.e., leave field blank)
LOGNORML - standard deviation
NORMAL - standard deviation
UNIFORM - range about the mean (i.e., the difference between the largest possible duration and the mean duration).
GAMMA - K
ERLANG - order of distribution function
WEIBULL - shape parameter

UNITS: The time units used in specifying the duration of the Action. The valid entries for this field and their meaning are as follows:

<u>Form entry</u>		<u>Meaning</u>
nseconds (ns)	-	nanoseconds
useconds (us)	-	microseconds
mseconds (ms)	-	milliseconds
seconds (s)	-	seconds
minutes (m)	-	minutes
hours (h)	-	hours
days (d)	-	days

The default value which is automatically placed in the form is SECONDS, but the user can change this default by using the Design User Interface UNITS command (see section 6.1.9).

COMMENT: Any user comment.

3.9.2 ALLOC

The ALLOC Primitive indicates the allocation of (request to use) a Resource which is needed by the Process. Whether a Resource requested by the ALLOC Primitive is actually obtained by a Process depends on a number of conditions, as described in the section on the Resource entity, section 3.6. If a Resource unit is in the idle state, it is available to be allocated to the requesting Process. If the Resource is busy, then allocated Resource units are checked to see if a Process can be preempted by priority (priority is inverse - priority 1 preempts priority 2) unless the Resource is protected with a LOCK primitive. The form for the ALLOC Primitive is shown in figure 3-14.

PARAMETERS FOR ALLOCATE:

ALLOCATE RESOURCE NAME:

NUMBER OF UNITS REQUESTED:

PARTIAL/ALL ALLOCATION:

ALLOCATION PRIORITY:

COMMENT:

Figure 3-14. Form for the ALLOC Primitive

Following is a description of the fields in the ALLOC form:

ALLOCATE RESOURCE NAME:	A reference to a Resource
NUMBER OF UNITS REQUESTED:	The number of Resource units to be allocated. 1 is the default.
PARTIAL/ALL ALLOCATION:	This specifies whether the Resource units will be allocated as they become available (PARTIAL) or only allocated simultaneously when they are all available (ALL). PARTIAL is the default.
ALLOCATION PRIORITY:	The priority to be used to determine which allocation request will be satisfied in the case of Resource contention. SPRIORITY is the default and evaluates to the priority of this Process.
COMMENT:	Any user comment.

3.9.3 ASSIGN

The ASSIGN Primitive is used to set the value of the following references:

1. a global Variable
2. a local (to the executing Process) variable
3. the attribute of an Item (currently attached to the Process)
4. the attribute of a Resource
5. \$CNODE (see section 3.17)
6. the attribute of a Process

Values that can be accessed for the assignment are:

1. signed, single precision, real or integer numbers
2. \$CLOCK (see section 3.17)
3. global Variables or Constants
4. local variables
5. Resources with any of the qualifiers NWAITQ, NBUSYQ, NINACTQ or NIDLEQ (see section 3.13)
6. Item attribute values
7. Queue qualifiers NQUEUE or TQUEUE (see section 3.13)
8. Resource attribute values
9. Process attribute values
10. an Item name
11. a Resource name
12. a Process name
13. a Queue name
14. a Table name
15. an Action name
16. \$NODE (see section 3.17)

17. \$NXTNODE (see section 3.17)
18. \$LINK (see section 3.17)
19. \$TASK (see section 3.17)
20. \$CNODE (see section 3.17)
21. an alpha literal (first character is \$) (see section 3.16)

The form for the ASSIGN Primitive is shown in figure 3-15.

```

PARAMETERS FOR ASSIGN

V1: [REDACTED] Q1: [REDACTED]

      TO

V2: [REDACTED] Q2: [REDACTED]

COMMENT: [REDACTED]

```

Figure 3-15. Form for the ASSIGN Primitive

In the form, V1 and Q1 are used to reference the current value, and V2 and Q2 are used to reference the value being set. For accessing values such as local variables, the simulation clock, etc., only the "V" fields need to be used. If the user is accessing an attribute of an entity, such as an Item, both the "V" and "Q" fields need to be used. The "V" field contains the name of the entity (Item, etc.) being accessed, and the "Q" field contains the name of the attribute of the entity whose value is desired or being set.

Following are examples of some typical entries:

V1: Item	V1: Item	V1: Variable
Q1: attribute	Q1: attribute	Q1:
V2: Item	V2: Variable	V2: Item
Q2: attribute	Q2:	Q2: attribute
V1: Variable	V1: Constant	V1: Constant
Q1:	Q1:	Q1:
V2: Variable	V2: Item	V2: Variable
Q2:	Q2: attribute	Q2:
COMMENT: Any user comment.		

Note that it is the entity specified by V2 and Q2 that takes on the new value specified by V1 and Q1.

3.9.4 BRANCH

The BRANCH Primitive indicates an unconditional branch to a named entry point. It is used for Process execution sequence control. The form for the BRANCH Primitive is shown in figure 3-16.

PARAMETERS FOR BRANCH:

BRANCH TO LABEL: XXXXXXXXXX

COMMENT: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Figure 3-16. Form for the BRANCH Primitive

Following is a description of the fields in the BRANCH form:

LABEL: The entry point to which the Process execution is to go
(which must be defined by an ENTRY Primitive).

COMMENT: Any user comment.

3.9.5 CALL

The CALL Primitive triggers execution of the called Process.

A CALL has one of three options (1) WAIT, (2) NOWAIT and (3) BLOCK. If a Process is called with the option WAIT, the calling Process will suspend execution until the called Process is completed. If a Process is called with the NOWAIT option, both called and calling Processes will execute simultaneously and will have no further communication. If a Process is called with the BLOCK option, the two Processes will execute in parallel until a WAIT Primitive is reached in the execution of the calling Process. When the WAIT Primitive is reached, the calling Process suspends execution until the called Process(es) complete(s). The principal purpose of the BLOCK option is to allow the calling of several different Processes, all of which must be completed before the calling Process will continue. If several Processes are called with the BLOCK parameter, the calling Process will suspend at the WAIT Primitive--whose presence somewhere below such a CALL Primitive is obligatory--until all of them have completed execution.

Two of the three kinds of Processes can be triggered via the CALL Primitive: parameter passing Processes and standard Processes. The triggering of an Item passing process is discussed in the section describing the SEND primitive. In triggering a parameter passing Process with a CALL Primitive, parameters are given to the called Process and/or parameters are returned to the calling Process. Parameters can be numerics, string literals, keywords, or the names of Items, Queues, Resources, Processes, Tables, and Actions. Parameter passing Processes with return parameters can be called only with the WAIT option. Standard Processes, which neither give nor return information may be called with any of the three options WAIT, NOWAIT and BLOCK.

The CALL also requires that a priority be established for the called Process. Priority is inverse, priority 1 preempts priority 2. This priority may be used by the called Process when competing with other Processes for available Resources (through the ALLOC Primitive with \$PRIORITY, see section 3.9.2).

The form for the CALL Primitive is shown in figure 3-17.

PARAMETERS FOR CALL

CALLED-PROCESS NAME: [REDACTED]

WAIT/NOWAIT/BLOCK: **NOWAIT** PRIORITY: [REDACTED]

GIVEN:

[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

RETURNS:

[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

COMMENT:

[REDACTED]

Figure 3-17. Form for the CALL Primitive

Following is a description of the fields in the CALL form:

CALLED-PROCESS NAME: The Process to be triggered.

WAIT/NOWAIT/BLOCK: Explained above. NOWAIT is the default.

PRIORITY: The priority associated with the triggered Process (discussed above).

GIVEN: Up to six parameters whose values are to be communicated to the called Process. Left blank if called Process is a standard Process.

RETURN: Up to six parameters whose values are to be returned to the calling Process. Left blank if called Process is a standard Process.

COMMENT: Any user comment

3.9.6 COMMENT

The COMMENT Primitive is used to add descriptive text to a Process. It has no effect on the operation of the Process. It is used simply to make the Process more understandable. A COMMENT Primitive can be placed anywhere within a Process after the START Primitive and before the END Primitive, and there is no limit to the number of COMMENT Primitives within a Process.

The form for the COMMENT Primitive is shown in figure 3-18

COMMENT:



Figure 3-18. Form for the COMMENT Primitive

Following is a description of the fields in the COMMENT form:

PARAMETERS FOR COMMENT: The fields provide the user with space to enter up to four lines (64 characters each) of descriptive text.

Note: In the simulation output report, each line of a COMMENT Primitive has an asterisk (*) appended to the beginning of the line. This asterisk is used by AISIM to recognize the COMMENT.

3.9.7 COMPARE

The COMPARE Primitive is used to model decisions based on user-controlled variables or the values of system keywords and attributes. The COMPARE performs the following operation:

IF P IS TRUE, THEN GO TO A

where:

"A" is an ENTRY label (defined by an ENTRY primitive) which is branched to if P is true.

"P" is a predicate which can be TRUE or FALSE. It consists of a phrase:

X1 OP X2

X1,X2 can be:

- (1) signed, single precision, real or integer numbers
- (2) global Variables or Constants
- (3) local Variables
- (4) Resources with either NWAITQ, NBUSYQ, NINACTQ or NIDLEQ attributes (which cannot be modified by the user) (see section 3.13)
- (5) \$CLOCK (see section 3.17)
- (6) a value specified by an Item name and attribute
- (7) a value specified by a Resource name and attribute
- (8) a value specified by a Process name and attribute
- (9) an Item name
- (10) a Resource name
- (11) a Process name
- (12) a Queue name
- (13) a Table name
- (14) an Action name
- (15) \$NODE (see section 3.17)
- (16) \$NEXTNODE (see section 3.17)

- (17) \$LINK (see section 3.17)
- (18) \$TASK (see section 3.17)
- (19) \$CNODE (see section 3.17)
- (20) an alpha literal (first character is \$) (see section 3.16)
- (21) a Queue with either NQUEUE or TQUEUE as an attribute (which cannot be modified by the user) (see section 3.13)

"OP" is one of the following operators:

- EQ - equal to,
- NE - not equal to,
- GE - greater than or equal to,
- GT - greater than,
- LE - less than or equal to,
- LT - less than.

Operation - "X1" is compared to "X2" using real, single precision arithmetic. If the comparison results in the same relation as "OP", then "P" is set TRUE and a branch is made to label "A"; otherwise, no branch is made (the next Process Primitive is executed).

The form for the COMPARE Primitive is shown in figure 3-19.

```

PARAMETERS FOR COMPARE

IF OPERAND 1: [REDACTED] QUALIFIER 1: [REDACTED]
RELATION: [REDACTED]
OPERAND 2: [REDACTED] QUALIFIER 2: [REDACTED]
BRANCH TO: [REDACTED]
COMMENT: [REDACTED]

```

Figure 3-19. Form for the COMPARE Primitive

The parameters of the form are filled in as indicated above.

3.9.8 CREATE

The CREATE Primitive is used to create Items (note the SEND Primitive can also create Items as part of its function). The initial attribute values (defined when the Item is declared) are assigned upon creation. Each Item created is attached to the Process. Two Items of the same name cannot exist in a Process at the same time. Item definitions are specified in the DUI. The form for the CREATE Primitive is shown in figure 3-20.

PARAMETERS FOR CREATE

ITEMS TO BE CREATED ARE:

Year	1990	1991	1992	1993	1994	1995
1990	1991	1992	1993	1994	1995	1996

COMMENT : [REDACTED]

Figure 3-20. Form for the CREATE Primitive

Following is a description of the fields in the CREATE form:

ITEMS: references to distinct Item types, instances of which are to be created

COMMENT: Any user comment.

3.9.9 DEALLOC

The DEALLOC Primitive indicates the release of previously allocated Resources. It is used to represent the release of a Resource (making it available to another request) upon completion of a job. The form for the DEALLOC Primitive is shown in figure 3-21.

PARAMETERS FOR DEALLOCATE:

DEALLOCATE RESOURCE NAME:

NUMBER OF UNITS DEALLOCATED:

COMMENT:

Figure 3-21. Form for the DEALLOC Primitive

Following is a description of the fields in the DEALLOC form:

RESOURCE NAME:	A reference to the Resource to be released.
NUMBER OF UNITS:	A reference to the integer number of Resource units to be returned to the idle state. 1 is the default.
COMMENT:	Any user comment.

3.9.10 DESTROY

The DESTROY Primitive is used to eliminate Items from the system, marking the end of the time in system. When an Item is destroyed, statistics on its time in the system are tabulated for the simulation's Item report.

The form for the DESTROY Primitive is shown in figure 3-22.

PARAMETERS FOR DESTROY

ITEMS TO BE DESTROYED ARE:

COMMENT: [REDACTED]

Figure 3-22. Form for the DESTROY Primitive

Following is a description of the fields in the DESTROY form:

ITEMS: References to distinct Item types, instances of which are to be destroyed.

COMMENT: Any user comment.

3.9.11 ENTRY

The ENTRY Primitive is used to define entry points from branches contained in the Primitives BRANCH, PROB, COMPARE, TEST, READ and LOOP. The form for the ENTRY Primitive is shown in figure 3-23.

PARAMETERS FOR ENTRY:

ENTRY LABEL: COMMENT: 

Figure 3-23. Form for the ENTRY Primitive

Following is a description of the fields in the ENTRY Primitive:

ENTRY LABEL: The 1-8 character name of the entry point used by the Primitive(s) which transfer control to it.

COMMENT: Any user comment.

3.9.12 EVAL

The EVAL Primitive is used to perform arithmetic functions within a Process so that model logic and timing can be a function of variables rather than a constant. This Primitive can also be used to access a Table. The EVAL Primitive is different from the other AISIM Primitives in that one of the EVAL parameters is a free-form expression whose result is assigned to the EVAL variable. The expression can be composed of references to several AISIM constructs within a single expression.

The valid functions that can appear in an expression are the following:

op1 + op2: add	- Returns the sum of op1 and op2
op1 - op2: subtract	- Returns the difference of op1 and op2
op1 * op2: multiply	- Returns the product of op1 and op2
op1 / op2: divide	- Returns the quotient of op1 and op2
op1**op2: exponentiation	- Returns the value of op1 raised to the power op2
absolute (op1)	- Returns the absolute value of op1
arccosine (op1)	- Returns the arc cosine of op1 in radians; $-1 \leq \text{op1} \leq 1$
arcsine (op1)	- Returns the arcsine of op1 in radians $-1 \leq \text{op1} \leq 1$
arctan (op1)	- Returns the arc tangent of (1/op1) in radians; op1 = (angle1/angle2); op1 < > 0.0
beta (op1, op2)	- Returns a random sample from a beta distribution op1 = power of x; op1 > 0 op2 = power of 1-x; op2 > 0
binomial (op1, op2)	- Returns a random sample from binomial distribution op1 = number of trials op2 = probability of success
cosine (op1)	- Returns the cosine of op1 in radians
Erlang (op1, op2)	- Returns a sample value from an Erlang distribution op1 = mean op2 = k (integer order of function)
exponent (op1)	- Returns a random sample from an exponential distribution op1 = mean
gamma (op1, op2)	- Returns a random sample from a gamma distribution op1 = mean op2 = k
integer (op1)	- Returns the integer part of op1
loge (op1)	- Returns the natural logarithm of op1 op1 > 0

lognorm1 (op1, op2)	- Returns a random sample from a normal distribution op1 = mean op2 = standard deviation
log10 (op1)	- Returns the common logarithm of op1 op1 > 0
normal (op1, op2)	- Returns a random sample from a normal distribution op1 = mean op2 = standard deviation
Poisson (op1)	- Returns a random sample from a Poisson distribution op1 = mean
random	- Returns a random number between zero and one
sine (op1)	- Returns the sine of op1 in radians op1 ≥ 0
sqrt (op1)	- Returns the square root of op1 op1 ≥ 0
tangent (op1)	- Returns the tangent of op1 in radians
uniform (op1, op2)	- Returns a uniformly distributed random sample between a range of values op1 = mean op2 = delta (the difference between the largest possible value and the mean value)
Weibull (op1, op2)	- Returns a sample value from a Weibull distribution op1 = shape parameter op2 = scale parameter

The operands (op1 and op2) for the above functions can consist of the following types of values:

- signed, single precision, real or integer numbers
- named Variable or Constant entities
- named local variables
- references to Process, Resource, or Item attributes
- references to Table entries
- \$CLOCK
- \$PRIORITY

The syntax for function calls in an expression is as it appears in the above list. The function name appears followed by its operands separated by commas and enclosed in parentheses. Function operands can be either the values listed above, or another expression consisting of functions using the above values. Spaces are ignored within an expression.

Following is a Bachus-Naur Form (BNF) type description of the syntax of valid arithmetic expressions. The symbol "|" is an "or" symbol, which means that any one of the choices may be selected. The symbol "::=" means that any of the choices on the right of the symbol can be used to replace the element named on the left of the symbol.

expression ::= expression addop term | addop term | term

term ::= term mulop factor | factor

factor ::= factor ** primary | primary

primary ::= function-call | (expression) | numeric-literal |
 entity-name [attribute] | Table-name(expression) |
 unqualified-entity

function-call ::= function (expression, expression) |
 function(expression) | function

addop ::= + | -

mulop ::= * | /

"function" is any function name appearing in the above list.

"Entity-name" is a named Process, Resource or Item entity, or a local variable which refers to the Process, Resource or Item.

"Table-name" is named Table entity or a local variable which refers to the Table.


"Unqualified entity" is a named global Variable or Constant, or a local variable.

All numeric calculations are performed in double precision, real arithmetic and then rounded to single precision values.

The form for the EVAL Primitive is shown in figure 3-24.

PARAMETERS FOR EVAL

SET VARIABLE: 

ARITHMETIC EXPRESSION:


COMMENT: 

Figure 3-24. Form for the EVAL Primitive

Following is a description of the fields in the EVAL form:

SET VARIABLE: The local variable or global Variable or Constant whose value is to be set.

ARITHMETIC EXPRESSION: Up to four lines for an expression in the valid syntax and using the types of variables described above.

COMMENT: Any user comment. Note: The comment is displayed on the Process graphics only if three or fewer lines are used for the arithmetic expression.

3.9.13 FILE

The FILE Primitive is used to place an Item on a user-defined Queue.

The effect of filing an Item on a user-defined Queue is to keep it in storage after the Process from which it is filed has ceased execution. The form for the FILE Primitive is shown in figure 3-25.

PARAMETERS FOR FILE:

FILE ITEM NAME: OPTION: ON QUEUE:
COMMENT:

Figure 3-25. Form for the FILE Primitive

Following is a description of the fields in the FILE form:

FILE ITEM NAME: A reference to the Item to be filed.

OPTION: The location in the Queue at which the entity is to be filed relative to the Queue pointer. The following can be used:

FIRST - The entity is placed first and the Queue pointer is set to it.

LAST - The entity is placed last and the Queue pointer is set to it. This is the default.

NEXT - The entity is placed after the current Queue pointer position in the Queue and the Queue pointer is reset to it.

BEFORE - The entity is placed before the current Queue pointer position in the Queue and the Queue pointer is reset to it.

QUEUE: The Queue on which the Item is to be filed.

COMMENT: Any user comment.

3.9.14 FIND

The FIND Primitive is used to reset the Queue pointer on a user-defined Queue (section 3.4) or a cross-reference set (section 3.5.2), and to assign to a local variable a "locator" pointer to a current position in the Queue. The rules governing Queue pointers are covered above in the section on user-defined Queues. The form for the FIND Primitive is shown in figure 3-26.

PARAMETERS FOR FIND:

FIND OPTION: NEXT ITEM NAME: ON QUEUE:

COMMENT : [REDACTED]

Figure 3-26. Form for the FIND Primitive

Following is a description of the fields in the FIND form:

FIND OPTION: The location (FIRST, LAST, NEXT, or BEFORE) of the Item or member of the cross-reference set to be assigned to the variable relative to the present Queue pointer. NEXT is the default.

ITEM NAME: The local variable which will refer to the Item or member of a cross-reference set.

ON QUEUE: The name of the Queue or cross-reference set that is to be traversed. If the cross-reference set is intended, the entity type whose cross-reference set is to be traversed is entered.

COMMENT: Any user comment.

The effect of locating an element with the FIND Primitive is (1) to set the Queue pointer to the beginning or end of the ordered holding area (i.e., FIRST or LAST) or relative to the previous location of the Queue pointer (i.e., NEXT or BEFORE), and (2) to assign the element in the position then indicated to the local variable.

3.9.15 LOCK

The LOCK Primitive prevents a Process from being suspended by losing Resources to a "higher" priority Process (priority is inverse, priority 1 preempts priority 2). LOCK is used to represent uninterruptible work. If LOCK is not used, Process execution can be suspended by a higher priority Process. When a Process loses any one of the Resources it has allocated it stops execution and is placed on a system-defined queue (the wait queue) until the Resource is again available. The LOCK Primitive overrides this suspension. The form for the LOCK Primitive is shown in figure 3-27.

PARAMETERS FOR LOCK:

COMMENT: 

Figure 3-27. Form for the LOCK Primitive

Following is a description of the field in the LOCK form:

COMMENT: Any user comment.

3.9.16 LOOP

The LOOP Primitive causes a branch to a named entry point a specified number of times. The form for the LOOP Primitive is shown in figure 3-28.

PARAMETERS FOR LOOP:


LOOP TO LABEL: LOOP  TIMESCOMMENT: 

Figure 3-28. Form for the LOOP Primitive

Following is a description of the fields in the LOOP form:

LABEL: The name of the ENTRY label (defined by an ENTRY Primitive) to which execution is to branch.

LOOP: Indicates the number of times Primitives between the ENTRY label and the LOOP Primitive will be executed. This includes the initial pass. For example, if 10 was used, then for each execution of the Process, the Primitives between the Entry label and the LOOP Primitive would be executed 10 times. Execution control would branch back to the ENTRY label 9 times.

COMMENT: Any user comment.

3.9.17 PROB

The PROB Primitive is used to model stochastic decision making. It indicates a probabilistic branch to a named entry point. Random number selection for the probabilistic branch can be controlled by the use of the EDIT STREAM command in the AUI. The form for the PROB Primitive is shown in figure 3-29.

PARAMETERS FOR PROBABILISTIC BRANCH:

BRANCH TO LABEL: PROBABILITY OF BRANCH: %COMMENT:

Figure 3-29. Form for the PROB Primitive

Following is a description of the fields in the PROB form:

- LABEL: The ENTRY label (defined by an ENTRY Primitive) to which the branching is to take place.
- PROBABILITY: The probability with which the branching is to take place, expressed in (integer) percent.
- COMMENT: Any user comment.

3.9.18 READ

The READ Primitive is used to obtain values from an external file for use within a model during a simulation run. This Primitive facilitates the use of case-study type simulation runs where there is a file for each case study containing values which will be read into the model for each run. In this way the data for the model can be changed without having to change the data in the model data base. The form for the READ Primitive is shown in figure 3-30.

PARAMETERS FOR READ:

```

READ FROM FILE: [REDACTED]
END OF FILE BRANCH LABEL: [REDACTED]
V1: [REDACTED] Q1: [REDACTED]
COMMENT: [REDACTED]

```

Figure 3-30. Form for the READ Primitive

Following is a description of the fields in the READ form:

READ FROM FILE: A 1 to 8 character logical name for the file to be read from. The default for the actual external file name associated with this logical name is "name.TXT", but the user can change the actual file name associated with this logical name (see section 3.11).

END OF FILE BRANCH LABEL: The ENTRY label (defined by an ENTRY Primitive) to be branched to in case an end-of-file is detected.

V1: The variable to be set to the value read from the file.

Q1: A qualifier for the entity in V1. For example if V1 is an Item reference, Q1 can be the name of an attribute of the Item whose value is to be set.

COMMENT: Any user comment.

The reference consisting of V1 and Q1 can be one of the following:

- a global variable
- a local variable
- the name of a Resource, Item, or Process for V1 and either the name of an attribute or "[]" for Q1. In the case of "[]" the READ Primitive will read in all attributes of the entity.
- the keyword \$CNODE

3.9.19 REMOVE

The REMOVE Primitive is used to remove an Item from a user-defined Queue.

The effect of removing an Item is to make it inaccessible to other Processes until it has been placed on another Queue (through the FILE Primitive) or delivered to another Process through the SEND Primitive. The form for the REMOVE Primitive is shown in figure 3-31.

PARAMETERS FOR REMOVE:

REMOVE OPTION: FIRST ITEM NAME: FROM QUEUE:

COMMENT: [REDACTED]

Figure 3-31. Form for the REMOVE Primitive

Following is a description of the fields in the REMOVE form:

REMOVE OPTION: The location in the Queue of the Item to be removed.
The option can be one of the following:

- FIRST - The first entity is removed and the Queue pointer is reset to the new first element. This is the default.
- LAST - The last entity is removed and the Queue pointer is reset to the new last element.
- NEXT - The entity associated with the current Queue pointer location is removed and the Queue pointer is reset to the succeeding element to it in the Queue.

ITEM NAME: The local variable that will contain the Item which is removed from the Queue. If there is no Item to be removed, this local variable is set to zero.

FROM QUEUE: The Queue from which the Item is to be removed.

COMMENT: Any user comment.

3.9.20 RESET

The RESET Primitive redefines the number of available units of a named Resource to plus or minus the indicated value. It is used to represent the increase or decrease of the available number of Resource units. The form for the RESET Primitive is shown in figure 3-32.

PARAMETERS FOR RESET:

RESOURCE NAME:
RESET BY (+/-) UNITS
COMMENT:

Figure 3-32. Form for the RESET Primitive

Following is a description of the fields in the RESET form:

RESOURCE:	A reference to a Resource whose available units are increasing or decreasing.
RESET BY (+/-):	The number of units to be added to or subtracted from those presently available. If more units are to be made available, this value is positive. If units are to be made unavailable, this value is negative.
COMMENT:	Any user comment.

3.9.21 RESUME

The RESUME Primitive is used to control explicitly the resumption of a Process which has been suspended through the SUSPEND Primitive. Resources deallocated at the time of suspension must be obtained again before Process execution progresses. The requests for these Resources is automatically handled by the RESUME Primitive. There is no preferential treatment given to these requests. They are treated in the same manner as an ALLOC Primitive. The form for the RESUME Primitive is shown in figure 3-33.

RESUME PROCESS REFERENCED BY

V1: [REDACTED] Q1: [REDACTED]

COMMENT: [REDACTED]

Figure 3-33. Form for the RESUME Primitive

The fields V1 and Q1 constitute a reference to task that is being resumed (see SUSPEND) and the COMMENT field is any user comment.

3.9.22 SEND

The SEND Primitive is used to send up to six Items to an Item passing Process. If an Item to be sent is not currently attached to the sending Process, it is automatically created. When the Items are sent, the receiving Process determines whether all the Items required by its definition have been received. If they have, the Process then initiates; if not, it will wait until all of the necessary Items have been received before executing. The form for the SEND Primitive is shown in figure 3-34.

PARAMETERS FOR SEND

SEND ITEMS TO [REDACTED]

ITEMS TO BE SENT ARE:

[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]

COMMENT: [REDACTED]

Figure 3-34. Form for the SEND Primitive

Following is a description of the fields in the SEND form.

- SEND: A reference to the Process to which Items are to be sent.
- ITEMS: References to up to six Item types, instances of which are to be sent.
- COMMENT: Any user comment.

3.9.23 SUSPEND

The SUSPEND Primitive is used to suspend the Process in which it appears. A Process that suspends itself with this Primitive may only be resumed by another Process which uses the RESUME Primitive. Since the RESUME Primitive must be able to refer to the task instant to be resumed, the suspending Process instance must save a reference to itself (i.e., assign the value of the keyword \$TASK to a global Variable or send it as an attribute of an Item) for later access by a RESUME Primitive. See section 3.17 for a description of \$TASK. The SUSPEND Primitive causes the deallocation of the Resources allocated to the Process. The form for the SUSPEND Primitive is shown in figure 3-35.

PARAMETERS FOR SUSPEND:

COMMENT: 

Figure 3-35. Form for the SUSPEND Primitive

Following is a description of the field in the SUSPEND form:

COMMENT: Any user comment.

3.9.24 TEST

The TEST Primitive indicates a branch to a named ENTRY Primitive if a Resource or Queue is not available. It is used to model decision making based on the availability of needed Resources or Queues. The form for the TEST Primitive is shown in figure 3-36.

PARAMETERS FOR TEST:


RESOURCE NAME: BRANCH TO LABEL  IF NOT AVAILABLECOMMENT: 

Figure 3-36. Form for the TEST Primitive

Following is a description of the fields in the TEST form:

RESOURCE NAME:	A reference to the Resource or Queue being tested for availability.
BRANCH TO LABEL:	The name of the ENTRY label (defined by an ENTRY Primitive) to which execution is to branch if the Queue or Resource is not available.
COMMENT:	Any user comment.

3.9.25 TRACE

The TRACE Primitive starts a debugging mechanism that is useful for analyzing the dynamics of an AISIM model. The effect of the TRACE Primitive is to create a file that records every execution of a Process and of the following Primitives within the Process.

START

CALL

ALLOC

DEALLOC

END

RESUME

RESET

SUSPEND

TRACE (on or off)

These Primitives are traced because they introduce major changes in the state of the system into a simulation run.

When the TRACE Primitive is operating, every instance of these Primitives in every Process is recorded either for the remainder of the simulation or until TRACE is turned off. The trace line writes out the simulation clock time, the node in which the Primitive is executed, and the Process executing the Primitive. The format for a trace line is the following:

T = "clock time" N = "node name" P = "Process name" "Primitive parameter"

The form for the TRACE Primitive is shown in figure 3-37.

PARAMETERS FOR TRACE

ON-OFF: ☐COMMENT:

Figure 3-37. Form for the TRACE Primitive

Following is a description of the fields on the TRACE form:

ON/OFF: "ON" to enable the TRACE.

"OFF" to disable the TRACE.

COMMENT: Any user comment.

3.9.26 UNLOCK

The UNLOCK Primitive cancels the effect of a previously executed LOCK Primitive. It is used to represent the conclusion of the uninterruptable phase of a Process. The form for the UNLOCK Primitive is shown in figure 3-38.

PARAMETERS FOR UNLOCK:

COMMENT: 

Figure 3-38. Form for the UNLOCK Primitive

Following is a description of the field in the UNLOCK Primitive:

COMMENT: Any user comment.

3.9.27 WAIT

The WAIT Primitive is used in conjunction with the CALL Primitive when the BLOCK option is used. The WAIT Primitive indicates that the calling Process is to be suspended until all Processes it triggered by a CALL with the BLOCK option have completed and returned control to the calling Process. It is generally used to model phenomena such as assembly points, executive schedulers, and other events in which progress cannot continue until several parallel activities are completed. Resources currently in possession of the calling Process are not deallocated. The form for the WAIT Primitive is shown in figure 3-39.

PARAMETERS FOR WAIT:

COMMENT: 

Figure 3-39. Form for the WAIT Primitive

Following is a description of the field in the WAIT Primitive:

COMMENT: Any user comment.

3.9.28 WRITE

The WRITE Primitive is used to write values from a simulation run to an external file. This data can be helpful in debugging or verifying a model. The form for the WRITE Primitive is shown in figure 3-40.

PARAMETERS FOR WRITE:

WRITE TO FILE: V1: Q1: COMMENT:

Figure 3-40. Form for the WRITE Primitive

Following is a description of the fields in the WRITE Primitive:

- WRITE TO FILE: A 1 to 8 character logical name for the file to which the data is written. The default for the actual external name associated with this logical name is "name.TXT", but the user can change the actual file name associated with this logical name (see section 3.11).
- V1: The local variable or entity reference whose value is to be written to the file.
- Q1: A qualifier for the entity in V1. For example if V1 is an Item reference, Q1 can be the name of an attribute of the Item whose value is to be written.
- COMMENT: Any user comment.

The reference consisting of V1 and Q1 can be one of the following:

- a global variable
- a local variable
- an attribute value by using the name of a Resource, Item, or Process for V1 and the name of an attribute or "[]" for Q1. In the case of "[]", the WRITE Primitive will write all of the attribute values to the file.
- a Process, Resource, Item, Queue, Table or Action, in which case the name of the entity is written.
- an alpha literal
- keywords \$NXTNODE, \$LINK, \$TASK, \$CNODE, \$PRIORTY, \$CLOCK
- Resource with NWAITQ, NBUSYQ, NINACTQ, or NIDLEQ as an attribute
- Queue with NQUEUE or TQUEUE as an attribute

LEGAL PATH TABLE

3.10 LEGAL PATH TABLE - NODE - LINK

The Legal Path Table (LPT) entity is the means by which the user can model physical communication paths between Resources. Typically, this is referred to as inter-node communication. When the LPT is not used, the communication mechanisms are implicit in the Process logic and do not usually have explicit Resources that cause communication queueing and transfer delays.

Two other model elements need to be discussed as part of the LPT entity; these are nodes and links. Nodes represent the points in an architecture where processing occurs. Links are the communication paths between nodes. Each node and link is actually a model Resource -- the name of the Resource being the name of the node or link. Full duplex links (denoted by "_F" after the link name) are two Resources. One will be named the name of the link with "_A" appended to it and the other with "_B".

The LPT consists of a four part list that specifies the FROM node, a TO node, a NEXT node, and a LINK. An example of Legal Path Table entries is given in figure 3-41.

FROM NODE	TO NODE	NEXT NODE	VIA LINK
=====	=====	=====	=====
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
D	C	C	C4
D	E	E	C6
D	F	F	C5
E	D	D	C8
E	G	G	C8
F	D	D	C5
F	G	G	C7
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

Figure 3-41. Sample Legal Path Table Entries

The headings indicate that to move from the FROM node to the TO node one must first go to the NEXT node via the LINK.

The LPT is a passive entity in that it does not contribute directly to the simulation statistics but, instead, is simply a table of values used by a model to effect data flow through a system. It is only changed through the Architecture Design Editor and therefore remains constant for any specific simulation run. Processes reference the LPT through the ASSIGN or COMPARE Primitives using \$CNODE (current node), \$NXTNODE (next node as specified in LPT), and \$LINK (the Link for the transfer) keywords (see section 3.17).

Operation - Every Process in AISIM can be set to execute in a specific node. Using the LPT through the keywords and the ASSIGN Primitive, a Process can locate itself in the network and reference other nodes. The referencing is done symbolically so that a Process can do this when executing. This allows AISIM to model different architectures without changing the model Processes.

3.11 FILES

The File entity is used to represent an external file which can be accessed during a simulation run. File entities are tied to READ and WRITE Primitives. Whenever a user places a READ or WRITE Primitive within a Process, the user specifies the logical name of a file that will be accessed during that read or write operation. A File entity is created with the same name as specified in the READ or WRITE Primitive for each unique logical file name. The user is not allowed to create, delete or copy File entities, or change the name of a File entity; AISIM automatically maintains all File entities. The user may edit a File entity with the DUI EDIT command to change its description (see section 6.1.4). The form for the File entity is shown in figure 3-42.

FILE: [REDACTED]

DESCRIPTION: [REDACTED]

Figure 3-42. Form for the File Entity

Following is a description of the fields in the File form:

FILE: 1 to 8 character logical file name

DESCRIPTION: Any user comment (0 to 53 characters)

Each File entity contains the logical name of a file to be accessed during the simulation. Associated with each logical file name, there must be an actual, physical file to which data is written or from which data is read. AISIM maintains a file called project.FNM that contains a list of all logical file names in a model and the physical files associated with each of those logical file names. During a Design session, AISIM reads in the information in the ".FNM" file for a model, makes updates as necessary as the user updates the model, and then rewrites the file at the end of the Design session.

Whenever the user uses a new logical file name in a model, the physical file assumed to be associated with that name is "logical.TXT". For example if a user places the name "PARMDATA" in a file name field of a READ Primitive, a File entity called PARMDATA is created, and the physical file to be associated with that logical name is set to be "PARMDATA.TXT". This association is then written out to the file project.FNM when the user exits the Design function.

When a simulation is run, it will read the project.FNM file to determine the names of all files to be accessed. All files to be read from must exist prior to the simulation, and files to be written to may or may not exist. If the user wishes to associate a file other than the default "logical.TXT" with a

logical file name, the user can edit the ".FNM" file and change the name of the physical file to be accessed. This type of change will facilitate the use of various case studies for a model. The physical file name in the ".FNM" file can be up to 43 characters in length.

TABLES

3.12 TABLES

Tables are user definable functions with 1 to 15 entries. Each entry consists of an X-VALUE and a Y-VALUE. The following may be used for these parameter values: (1) both numeric, (2) one numeric and the other alphanumeric or (3) both alphanumeric.

Tables are accessed by using the EVAL Primitive. In the EVAL Primitive expression field, the Table name is followed by an X-VALUE enclosed in parentheses. The SET VARIABLE will be set to the Y-VALUE which maps from the X-VALUE. A Table evaluation can also be part of a more complex arithmetic expression as long as the Y-VALUE is numeric.

3.12.1 Discrete Tables

If the Table accessed is discrete (TYPE is D), the Table entry's X-VALUE must be numeric, and the X-VALUE entries must be in increasing order. The Y-VALUE extracted from the Table is that value associated with the X-VALUE that is equal to or less than the X-VALUE given in the Table index. For example, if an X-VALUE of 3.5 is used as the index and the nearest X-VALUES in the Table are 3 and 4, the Y-VALUE associated with the X-VALUE of 3 will be extracted and placed in the given SET VARIABLE name. If the index is less than the smallest X-VALUE, the value returned is the Y-VALUE associated with the largest X-VALUE.

3.12.2 Continuous Tables

If the Table accessed is continuous (TYPE is C), all X-VALUE and Y-VALUE entries must be numeric. The SET VARIABLE of EVAL is set by the following rules :

- a. the Y-VALUE associated with the X-VALUE that equals the index, or
- b. the interpolation of the Y-VALUE associated with the X-VALUE which is less than the index and the X-VALUE greater-than the index, or
- c. the Y-VALUE associated with the largest X-VALUE, if no interpolation is possible.

3.12.3 Alphanumeric Tables

If the Table is defined as alphanumeric (TYPE is A), one or both X-VALUE and Y-VALUE for each entry must be a name of a model entity. The SET VARIABLE is set to the Y-VALUE corresponding to the X-VALUE.

If the expression for an index into a Table in the EVAL Primitive does not correspond to an X-VALUE in the Table referenced, an execution error message will be printed in the analyze report and the value of the SET VARIABLE will remain unchanged.

The form for the Table entity is shown in figure 3-43.

TABLE:		TYPE:	
COMMENT:			
X VALUE	Y VALUE		

Figure 3-43. Form for the Table Entity

Following is a description of the fields in the Table form:

TABLE:	1 to 8 character name of table
TYPE:	C - continuous, D - discrete or A - alphanumeric.
X VALUE:	x-axis value
Y VALUE:	y-axis value
COMMENT:	any user comment. (0 to 53 characters)

Table entities are entered using the Design User Interface EDIT command (see section 6.1.4).

ATTRIBUTES

3.13 ATTRIBUTES

Certain AISIM constructs have associated attributes which can take as values, (1) numerics, (2) alpha literals (3) entity names, or (4) keywords. Some attributes are user defined. Others are dynamic attributes which are recognized and modified by the AISIM simulator.

The values of attributes may be accessed by a Process with the ASSIGN and COMPARE Primitives. The forms for both of these Primitives use two fields to indicate the value accessed. The first field contains the name of the entity and the second the name of an attribute associated with it.

Three AISIM entities, Processes, Resources and Items, may have attributes specified by the user. These attributes allow the modeler to define a unique set of characteristics for certain entities. An example is a channel. Channels have a physical attribute of maximum transfer rate. This characteristic is assigned to the AISIM Resource by specifying an attribute of RATE for the channel Resource.

Simulation experience has shown that some logic in a system is dependent on the system's dynamics. That is, some activity is dependent on queue lengths or the number of busy Resources. Since this phenomenon is fairly common, AISIM has embedded features to model this. The following attributes are built into the AISIM simulator for each instance of an entity. These attributes may be accessed by the COMPARE and ASSIGN Primitives, but the values for the Resource and Queue attributes may not be changed by the user.

Entity	Attribute	Description
Resource	NIDLEQ	the number of units of the Resource which are in an idle state
	NBUSYQ	the number of units of the Resource which are in a busy state
	NINACTQ	the number of units of the Resource which are in an inactive state
	NWAITQ	the number of Processes executing which are waiting for a Resource unit to be deallocated
Item	TAIL	the sequential creation number of the Item
	PRIORITY	the priority of the Item
Queue	NQUEUE	the number in the Queue
	TQUEUE	the average time entities are in the Queue

CONSTANTS AND VARIABLES

3.14 CONSTANTS AND GLOBAL VARIABLES

Constants and Variables are entities used to define global parameters of a model, that is, values which may be accessed by all Processes. There is an implicit caution which must be used when using these entities. Because AISIM simulates multi-processing, global parameters can be accessed "concurrently" by more than one Process. Care should be taken when multiple Processes modify the same global Variable.

A Constant is given a numeric value before the start of a simulation. The value must be numeric and can not be changed by the simulation. A Variable may be set to (1) an alpha literal, (2) the value of a keyword, or (3) to any other AISIM entity that may be accessed by the EVAL and ASSIGN Primitives. A Variable's value may vary throughout the simulation.

The initial values of both Constants and Variables are set in the Design portion of AISIM. The value of both entities may be reset, before the simulation is started, in the Analysis function.

While the value of a Constant may not be changed during the simulation, the initial value of a Variable may be changed by the user (between periods or at break points) or by the model itself (by use of the ASSIGN and EVAL Primitives).

Constants and Variables may be used in place of a numeric value anywhere a numeric value is required with the following exceptions:

1. The number of units of a Resource may only be a Constant or a numeric value.
2. The initial value of a Constant must be a numeric value.

The forms for Constants and Variables are shown in figure 3-44.

CONSTANT:	
VALUE:	
DESCRIPTION:	
HF1-HELE:	
VALUE:	
DESCRIPTION:	

Figure 3-44. Forms for Constant and Variable Entities.

Following is a description of the fields in the Constant and Variable forms:

VARIABLE/CONSTANT: 1 to 8 character name of Variable or
Constant.

VALUE: 8 digit floating point or any AISIM variable
reference to a numeric value.

DESCRIPTION: Any user comment. (0 to 53 characters)

Constant and Variable entities are defined using the Design User Interface
EDIT command (see section 6.1.4).

LOCAL VARIABLES

3.15 LOCAL VARIABLES

AISIM has two kinds of variables: local and global. Global Variables are those explicitly defined within the model and given initial values. Local variables are variables that appear in Process Primitives but are not otherwise defined. Local variables enable Processes to execute in parallel without interfering with each other because each Process has an independent set.

At the beginning of the execution of a Process all local variables are initialized to zero. They will remain so unless other values are explicitly assigned to them. Local variables may be assigned values with the ASSIGN and EVAL Primitives or through parameter passing. Local variables may be assigned the following values:

Numeric - a floating point or integer number

Global Constant or Global Variable value

Another local variable

A Resource name

A Process name

An Item name

A Queue name

An Action name

A Table name

An alpha literal (first character \$)

The value of a keyword evaluation

Although "local," the values of such variables can be communicated from one Process to another through parameter passing (i.e., through the CALL Primitive). Note that logical file names within READ and WRITE Primitives cannot be passed in via a CALL Primitive. Local variables can be used to fill in any parameter slot in any Primitive that is not an option, a label, a distribution or function, and including:

Item attribute

Resource attribute

Process attribute

CALL given parameter

CALL return parameter
Process given parameter
Process return parameter
ALLOC Resource name
DEALLOC Resource name
CALL Process name
CALL Priority name
ASSIGN set variable (variable 2)
COMPARE variable
FILE Queue name
FILE Item name
FIND Queue name
FIND Item name
REMOVE Queue name
REMOVE Item name
RESUME task reference

ALPHA LITERALS

3.16 ALPHA LITERALS

An alpha literal is a character string. It consists of a \$ followed by up to seven other characters, as in

\$WAIT

and

\$JONES

that do not make up the name of a keyword (see next section). Alpha literals can be used to compare strings for identity or nonidentity with the COMPARE Primitive. They can be used as attributes. This is useful for making AISIM models more readable.

KEYWORDS

3.17 KEYWORDS

The following keywords are defined in the AISIM simulator and may be used in Process logic in any Primitive in which the evaluation of the keyword results in a value which is correct in context.

Like alpha literals, these terms begin with the character "\$". However, keywords function differently from alpha literals. Keywords evaluate to a value. In that sense they can be considered intrinsic functions.

\$CLOCK - The value of the current simulation clock during the execution of a simulation run. This keyword may be placed in any field of a Process Primitive which may contain a numerical value.

\$CNODE - The reference to the current node in which a Process is executing. All Processes can be set to execute in a node in the architecture. The node corresponds to a Resource. This keyword evaluates to the Resource. This keyword allows a modeler to control allocation and deallocation of a node from within the execution of a Process. This keyword can be assigned a value. This, in effect, changes the node in which a Process is logically executing. This is the only keyword that may be assigned a value in the Process logic.

\$TASK - The current instance of the Process in which this keyword appears. A Process executing in a simulation can assign the value of the \$TASK keyword to a global Variable. This allows one Process to suspend itself and another Process to resume it by referencing the Process to be resumed with the stored value of \$TASK.

\$PRIORITY - The priority of the currently executing Process. This keyword is generally used in an ALLOC Primitive to resolve Resource contention issues.

\$NODE - \$NODE takes one argument, a reference to a Process. Given a Process, \$NODE evaluates to the name of the node in which the Process has been defined to execute. This is the name of a Resource. This keyword allows a Process in AISIM to determine a destination for messages which request a specific Process to be executed. The node specification for a Process is defined by a user and is associated with the START symbol for the Process.

The following keywords directly access the legal path table and architecture structure. Each keyword evaluates to the name of a node or link Resource.

\$NXTNODE - \$NXTNODE takes one argument, a reference to a destination node. Given a destination node, \$NXTNODE assumes the current node (\$CNODE) of the executing Process is the source (FROM) node. Accessing the legal path table, \$NXTNODE returns the name of the next node along the path to the destination node. This is the name of a Resource. This keyword allows the AISIM modeler to write Processes that perform message forwarding through a network.

\$LINK - \$LINK takes one argument, a reference to a destination node. Given a destination node, \$LINK assumes the current node (\$CNODE) of the Process is the source (FROM) node. Accessing the legal path table, \$LINK evaluates to the name of the link to the next node along the path to the destination node. This is the name of a Resource.

MESSAGE ROUTING SUBMODEL

3.18 MESSAGE ROUTING SUBMODEL

When one Process triggers another through a CALL Primitive, the called Process is initiated in the same node as the calling Process. This is implicit in the AISIM simulator and is true even if the called Process is associated with a different node.

In order to model the functional distribution of Processes throughout a network, a logical Process communication feature had to be incorporated into AISIM. One requirement for this feature is that the delays inherent in the network communications be accurately represented in the model so that if a Process resident in one node initiates a Process resident in another node, the delays and queueing effecting this communication are taken into account. Also, AISIM is required to enable the analysis of different architectures performing the same functions with a minimum of change to the model.

To satisfy these requirements a special submodel has been devised to represent the routing of messages through an AISIM architecture and to initiate remote Process triggering. Since different protocols for network communication are conceivable, the AISIM message routing function has been implemented as an AISIM model and included in the AISIM system library under the name COMMUN-B. This enables an AISIM user to select and merge this model into his own. The advantage of this approach is that the user can review the logic in this submodel, determine its appropriateness to his problem and modify the message routing submodel if necessary. This will not often be the case because the message routing submodel applies to many communications networks.

The message routing submodel uses the architecture and Legal Path Table of a model through the use of the system-defined keywords and the Process Primitives.

The message routing submodel consists of one Item representing the message dispatched through the system architecture, four Processes representing the activities required for the inter-node communication and other supporting entities. Everything required for this model is included in the AISIM system library and can be merged into a user's model in a simple operation. (See section 10.2 of the Library User Interface.)

Additional details on the message routing submodel are provided in appendix D.

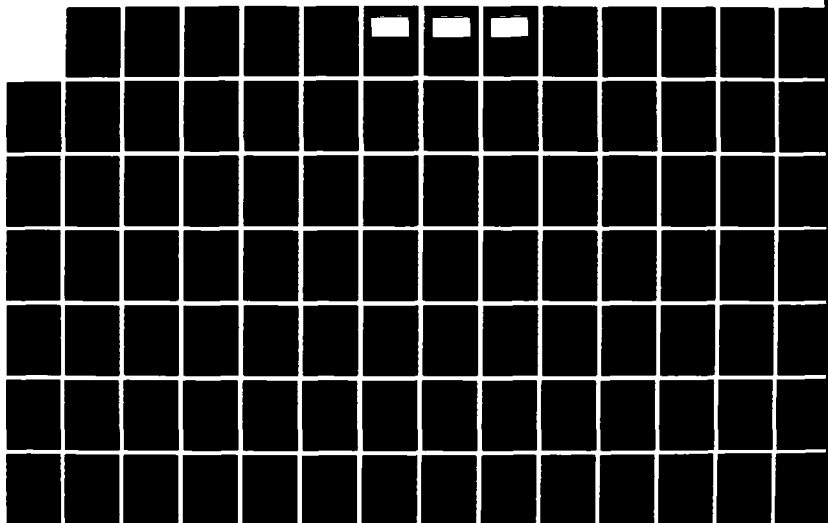
AD-A190 171

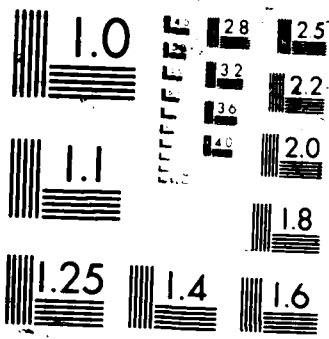
AUTOMATED INTERACTIVE SIMULATION MODEL (AISM) VAX
VERSION 50 USER'S MANU. (U) HUGHES AIRCRAFT CO
FULLERTON CA GROUND SYSTEMS GROUP V ALLERTON ET AL.
29 MAY 87 ESO-TR-87-228 F19628-86-C-0070 F/O 12/5

2/4

UNCLASSIFIED

NL





SECTION 4

AISIM SYSTEM OVERVIEW AND SYSTEM INITIALIZATION

The AISIM user interface consists of the following levels of operation:

- Level 1 - Not connected level
- Level 2 - VAX/VMS Ready level
- Level 3 - AISIM READY level
- Level 4 -
 - Level 4A - Design User Interface (DUI) Sublevel
 - Level 4B - Analysis User Interface (AUI) Sublevel
 - Level 4C - Replot User Interface (RUI) Sublevel
 - Level 4D - Hardcopy User Interface (HUI) Sublevel
 - Level 4E - Library User Interface (LUI) Sublevel
 - Level 4F - File Management User Interface (FUI) Sublevel
 - Level 4G - Help Editor Interface (HEI) Sublevel
- Level 5 -
 - Level 5A1 - Process Editor Interface (PEI) Sublevel
 - Level 5A2 - Architecture Design Editor (ADE) Sublevel
 - Level 5E1 - Mergein (MI)
 - Level 5E2 - Mergeout (MO)
 - Level 5E3 - Checkin (CI)
 - Level 5E4 - Checkout (CO)
 - Level 5E5 - Convert (CONV)
 - Level 5G1 - Update (UPD)

The relationship of these different levels is shown in figure 4-1. The current level of operation determines the system's response to a given command. For example, the command EDIT LOAD is valid only in the DUI level. Each level prompts the user for input with a specific symbol or phrase. For example, the AISIM READY level prompts with the phrase "AISIM READY" on the screen when it expects a command to be entered from the keyboard. The DUI level, on the other hand, prompts with an "**". The prompt for each level is shown in the figure in its box. The commands used to go from one level to another are shown next to the arrows indicating the direction of transfer.

4.1 REACHING THE AISIM READY LEVEL

The procedure for logging on is specific to a given computer system and the user is referred to local references for gaining access to the top level of the system on which AISIM is hosted. (This section assumes a VAX compatible host. For other installations please refer to installation specific instructions.) When prompted with:

\$

the user has reached Level 2 of AISIM operation. To reach Level 3, the user enters the command:

AISIM

When execution of this command completes, an audible 'beep' will be heard at the terminal and the AISIM READY prompt will appear on the terminal.

4.2 ACCESSING AISIM HELP

The help capability within AISIM provides a means for the user to receive and disseminate information pertinent to the AISIM tool. The help information is located in a separate help data base. This help information initially falls under one of two top level topics, modeling concept or function level. The modeling concept help information available is essentially that information found in chapter 3 of this manual. For each modeling concept that has subsections in chapter 3 a corresponding subtopic of the concept will be available within the help database. The other major topic within the help database is the function level topic. Help information on each AISIM function appearing in this manual is available within this topic. Within each function name subtopics corresponding to each function's valid commands will be available. Accessing these subtopics will provide help information on the use of that function's commands. In addition, subtopics corresponding to each command's parameters will provide further help information.

The structure of the help database supplied is as shown in figure 4-2.

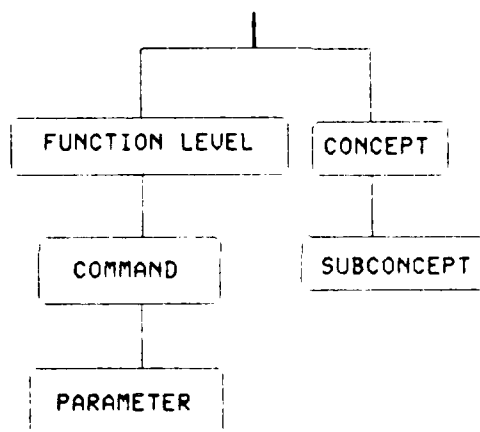


Figure 4-2. Help Database Structure

An extended capability of the AISIM help facility is the ability to create help information. An AISIM user may create or modify help messages through the Update function within the Help Editor Interface. Help information may be placed under one of three general top level topics: guideline, note or procedure. Any AISIM user may create or modify the information under these topics, however, only one user at a time may access the database for updating. The forms supplied for creating and modifying help messages are shown in figures 4-3 through 4-5.

GUIDELINE NAME: [REDACTED]

DATE: [REDACTED]

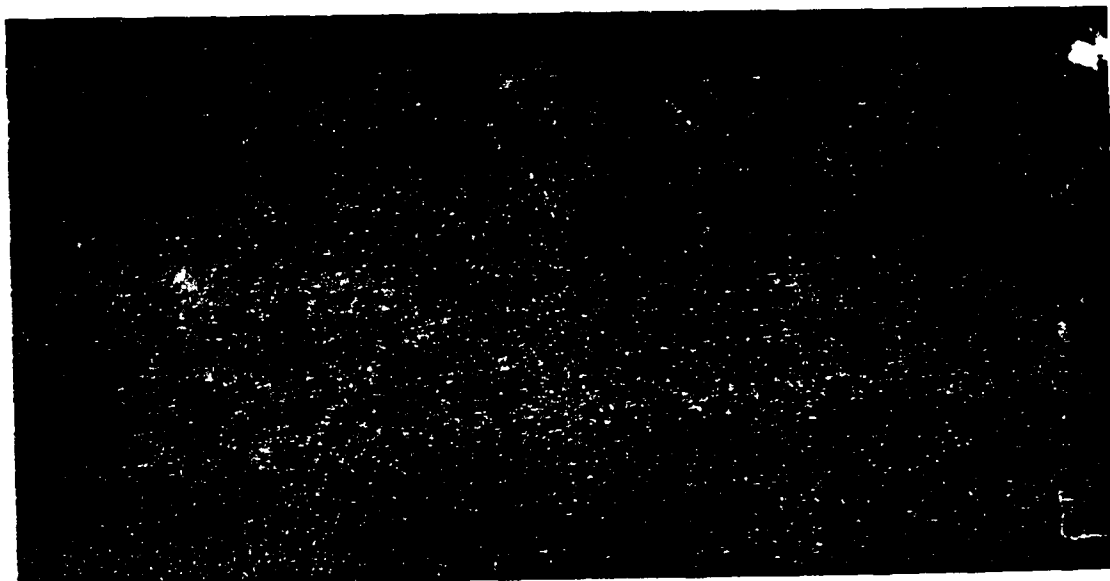


Figure 4-3. Form for the Guideline Help Topic

Following is a description of the fields in the Guideline form:

GUIDELINE NAME is the name of the guideline help topic.

DATE is an optional user supplied date.

NOTE NAME: [REDACTED]

DATE: [REDACTED]



Figure 4-4. Form for the Note Help Topic

Following is a description of the fields in the Note form:

NOTE NAME is the name of the note help topic.

DATE is an optional user supplied date.

PROCEDURE NAME: [REDACTED]

DATE: [REDACTED]

Figure 4-5. Form for the Procedure Help Topic

Following is a description of the fields in the Procedure form:

PROCEDURE NAME is the name of the procedure help topic.

DATE is an optional user supplied date.

A valid topic name will consist of up to 16 characters from the set of printable characters.

The date field can contain up to 20 characters.

The textual input will be assumed to end upon the detection of two blank lines in the form. This will allow the use of a single blank line for separation of text and will eliminate the display of trailing blank lines when the help information is later displayed. Up to 20 lines of 79 characters each of text may be input.

SECTION 5

AISIM READY LEVEL

At the AISIM READY level of operation a number of commands are available to the user for directing the course of the session (ANALYZE, CHANGE, BATCH, DESIGN, END), for manipulating the database (BACKUP, EDIT, RESTORE, LIBRARY), for requesting information about AISIM operation (HELP), for requesting model data (PRINT, HCOPY, GENLIST), for creating a data file to be used by a READ Primitive during a simulation run (FILMAN), for adding information to the available HELP messages (HEI), and for deleting temporary AISIM files (DELFILE). These commands are summarized in the command summary in figure 5-1 and described in the following sections. These commands may be entered only while in the AISIM READY level of operation (i.e., when the user has received an AISIM READY prompt).


```

ANALYZE [PROJECT(project)] [NOXLATE] [TERM(terminal)]
A      [P(project)]      [N]      [T(terminal)]

BACKUP [PROJECT(project)]
      [P(project)]

BATCH [TERM(terminal)]
      [T(terminal)]

CHANGE [PROJECT(project)] [TERM(terminal)]
C      [P(project)]      [T(terminal)]

DELFILE [PROJECT(PROJECT)]
DELF    [P(project)]

DESIGN [PROJECT(project)] [TERM(terminal)]
D      [P(project)]      [T(terminal)]

EDIT   [PROJECT(project)] [TRACE]
      [P(project)]

END

FILMAN FILE(filename) [ERRCHK(project)] [TERM(terminal)]
F      F(filename)    [E(project)]      [T(terminal)]

GENLIST [PROJECT(project)] [TERM(terminal)]
GLIST   [P(project)]      [T(terminal)]

HCOPY [PROJECT(project)] [TERM(terminal)]
HC     [P(project)]      [T(terminal)]

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

HEI [TERM(terminal)]
HEI [T(terminal)]

LIBRARY
LIB

LIST
L

LISTOFF

LISTON

MSGOFF

```

Figure 5-1. AISIM READY Level Command Summary

MSGON

PRINT [PRINT(project)]

P [P(project)]

REPLOT [PROJECT(project)] [TERM(terminal)]

RP [P(project)] [T(terminal)]

RESTORE [PROJECT(project)]

[P(project)]

Figure 5-1. AISIM READY Level Command Summary (cont'd)

5.1 INITIATING AN ANALYSIS SESSION

Simulation of the model developed under the DUI sublevel (see section 5.6) is accomplished through commands available in the AUI sublevel. The AUI is accessed from the AISIM READY level by issuing the following command:

```
ANALYZE [PROJECT(project)] [NOXLATE] [TERM(terminal)]
```

```
A [P(project)] [N] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database to be used. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[NOXLATE] is an optional parameter indicating that, FOR THIS ANALYSIS SESSION ONLY, no translation from the "project" database is to be performed, and simulation input from a previous translation is to be used. The "previous translation" must have been performed. If this parameter is omitted, the translation will be performed.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK   - TEK4105 terminal
VT    - VT100 terminal with Selanar graphics
```

The system will respond with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:    PRODUCTION VERSION 5.0
TERMINAL:   Terminal type specified in command or default
PROJECT:    Project specified in command or default
USER:       Userid
XLATE/NOXLATE: XLATE/NOXLATE, depending upon command.
ENTER YES TO PROCEED, NO TO ABORT...
```

Typing "yes" will cause the system to complete the transfer to the AUI sublevel. A "beep" will be given at the terminal and the AUI prompt (#) will appear when the system is ready to accept commands at the AUI sublevel. These commands are discussed in section 7. A "no" response will abort the command.

5.2 BACKING UP A DATABASE

To provide a backup of a project database, especially useful for saving a copy of the present model design before it is altered or modified, enter the following command:

```
BACKUP [PROJECT(project)]
```

```
BACKUP [P(project)]
```

where:

[PROJECT(project)] is an optional Parameter indicating the project database to be backed up. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

The system responds with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL:  Default terminal type
PROJECT:   Project specified in command or default
USER:      Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause a backup copy of the project to be stored in a database file named project.BCK. A "no" response will abort the command.

5.3 RUNNING AN ANALYZE SESSION VIA BATCH MODE

An analyze session can be submitted to run in Batch mode. The syntax for the BATCH command is as follows:

```
BATCH [TERM(terminal)]
      [T(terminal)]
```

where:

[TERM(terminal)] is an optional parameter indicating the new terminal type to be used. If omitted, the terminal type default value remains unchanged. The valid terminal types are the following:

HP	- HP2647A or HP2648A terminal
HP23	- HP2623 terminal
TEK	- TEK4105 terminal
VT	- VT100 terminal with Selanar graphics

This command causes a command file to be created which will run an analyze session. This command file can then be submitted to a batch queue on the user's system in order to run the simulation.

The advantages of the batch method are:

- 1) The user does not have to remain at the terminal through out the AISIM session. All necessary job information is specified up front and the system takes charge.
- 2) It is not necessary to use a graphics terminal. Any terminal connected to the VAX will suffice.
- 3) Multiple simulation runs can execute concurrently.
- 4) Simulation runs can be deferred to execute during off-peak hours.

The system prompts the user for the following information.

ENTER NAME OF PROJECT (1-8 char): the name of the project to be used.

DO YOU WISH TO TRANSLATE THE MODEL? yes or no based on the user's choice.

DO YOU WANT TO ADD A DESCRIPTION FOR THIS RUN? (Y/N) yes or no based on the user's choice. Yes causes a description form to be presented.

ENTER COMMANDS FOR AISIM RUN (<CR> TO END) >

Enter commands for AISIM run. Allowable AUI commands are CANBREAK, DELETE, EDIT, END, GET DEF, GO, INPRES, SAVE, and UNITS.

Commands are typed one per line, in the order they are to be acted upon. Commands must be typed in the correct format. The GO and END commands are mandatory. All other commands are optional.

After the above processing is completed, a file called SUBBATCH.COM will have been created. This file can then be submitted to an appropriate batch queue with any other information such as at what time the job should run (see VAX SUBMIT command for available parameters). If no extra information is necessary, the following command will submit the AISIM job to the default batch queue to be run immediately:

```
SUBMIT SUBBATCH.COM
```

Figures 5-2 and 5-3 show sample batch run setups.

```
AISIM READY
>BATCH
ENTER NAME OF PROJECT (1-8 CHAR): project
DO YOU WISH TO TRANSLATE THE MODEL? yes
DO YOU WANT TO ADD A DESCRIPTION FOR THIS RUN? (Y/N): yes
ENTER COMMANDS FOR AISIM RUN (<CR> TO END)
>e c,msgrate,1
>e c,errorate,2
>go
>end
>
SUBBATCH.COM CREATED
AISIM READY
>submit subbatch.com
    Job 303 entered on queue SYS$BATCH
AISIM READY
```

Figure 5-2. Sample Batch Job Submission

```

AISIM READY
>BATCH
ENTER NAME OF PROJECT (1-8 CHAR): project
DO YOU WISH TO TRANSLATE THE MODEL? yes
DO YOU WANT TO ADD A DESCRIPTION FOR THIS RUN? (Y/N): yes
ENTER COMMANDS FOR AISIM RUN (<CR> TO END)
>e c,msgrate,1
>e c,errorate,2
>get def,plotdef
>go
>save plot,plots,plots created in simulation run
>end
>no
>
SUBBATCH.COM CREATED
AISIM READY
>submit subbatch.com
    Job 304 entered on queue SYS$BATCH
AISIM READY

```

Figure 5-3. Sample Batch Job Submission with Plots

5.4 CHANGING THE CURRENT PARAMETERS

The current parameters of an AISIM session (PROJECT and TERMINAL) can be changed via the CHANGE command. The syntax for the CHANGE command is as follows:

```
CHANGE [PROJECT(project)] [TERM(terminal)]
```

```
C [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the new project database to be used. If omitted, the project default value remains unchanged.

[TERM(terminal)] is an optional parameter indicating the new terminal type to be used. If omitted, the terminal type default value remains unchanged. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal with Selanar graphics
```

This command causes the current default project and terminal to be set to the names entered. The current default parameters are then listed as follows:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL:  Default terminal type
PROJECT:   Project specified in command or default
USER:      Userid
```


5.5 DELETING PROJECT FILES

The DELFILE command is used to delete the following five files for a specified project:

- 1) project.XLT
- 2) project.WDB
- 3) project.RPT
- 4) project.LST
- 5) project.TRC

To delete these files, the user types:

```
DELFILE [PROJECT(project)]
```

```
DELF    [P(project)]
```

where:

[PROJECT(project)] is an optional parameter specifying the project name for the files. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

The system responds with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL: Default terminal type
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the files to be deleted. A "no" response will abort the command.

5.6 INITIATING A DESIGN SESSION

A project database is created/modified using the commands available in the DUI. The DUI is accessed from the AISIM READY level by issuing the following command:

```
DESIGN [PROJECT(project)] [TERM(terminal)]
```

```
D [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating that the desired project file to be acted upon by the command is "project", where "project" is a standard alphanumeric file label containing 1-8 characters beginning with an alpha character and containing no special characters or embedded blanks.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK   - TEK4105 terminal
VT    - VT100 terminal with Selanar graphics
```

The following is displayed after entering this command:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL:  Terminal type specified in the command or default
PROJECT:   Project specified in the command or default
USER:      Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response causes the completion of the level transfer. A "no" response will abort the command. If "yes", the terminal will display:

```
CREATING WORKING DATABASE.....
```

Followed by:

```
.....COPY COMPLETE
```

The DUI prompt (*) will appear when the system is ready to accept commands at the DUI sublevel. These commands are discussed in section 6.

The project database is stored in a database file named project.DBF. The working copy of the database is stored in a database file named project.WDB. The file association list is stored in a file named project.FNM.

5.7 VIEWING OUTPUT REPORTS

To access the model simulation report or model trace interactively on the terminal (via the EDT editor), enter the following command:

```
EDIT [PROJECT(project)]
```

```
EDIT [P(project)]
```

or

```
EDIT [PROJECT(project)] [TRACE]
```

```
EDIT [P(project)] [TRACE]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database to edit. If omitted, the project is assumed to be the last project specified in a previous AISIM READY command.

[TRACE] is an optional parameter indicating that the project trace file is desired for editing rather than the report file.

Result:

The EDT editor is entered with the file to be edited set according to the project. All EDT editor commands can be used on this file. The file is either the project report file (this is the default) or the project trace file. See section 13.3 for a brief discussion of relevant EDT text editor commands.

AISIM READY / END

5.8 RETURNING TO VAX/VMS READY LEVEL

To return to the VAX/VMS Ready level from the AISIM READY level, the user types the command:

END

The system will return to the VAX/VMS Ready Level and the screen will display

\$

5.9 CREATING AND EDITING AN INPUT FILE FOR THE READ PRIMITIVE

The FILMAN command is used to invoke the File Management User Interface (FUI) in order to create or edit a file which will be used during a simulation run as input to the simulation. The FUI is accessed from the AISIM READY level by issuing the following command:

```
FILMAN FILE(filename) [ERRCHK(project)] [TERM(terminal)]
```

```
F F(filename) [E(project)] [T(terminal)]
```

where:

FILE(filename) is a mandatory parameter indicating the name of the file to be used during the current FUI session.

[ERRCHK(project)] is an optional parameter indicating a database to be used to error check data being entered into the file.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP	- HP2647A or HP26478A terminal
HP23	- HP2623 terminal
TEK	- TEK4105 terminal
VT	- VT100 terminal with Selanar graphics

The following is displayed after this command is entered:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:      PRODUCTION VERSION 5.0
TERMINAL:     Terminal type specified in command or default
FILE:         File specified in command
ERRCHK:       Project used for error checking
USER:         userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the FUI to be invoked. The FUI prompt (*) will appear when the system is ready to accept commands. These commands are discussed in section 12. A "no" response will abort the command.

The file data is stored in a file called filename.TXT. The error check project is a design database called project.DBF.

5.10 CREATING A MODEL LISTING

The GENLIST command is used to produce a listing of a model without having to enter the AUI level and perform a complete translation of the model. The listing is identical to the Initialization Report section of the output report (see the section on AISIM Simulation Results Reporting). Elements of this report are:

- 1) Global Constant Definition
- 2) File Definition
- 3) Table Definition
- 4) Global Variable Definition
- 5) Item Definition
- 6) Queue Definition
- 7) Resource Definition
- 8) Architecture Legal Path Definition
- 9) Action Definition
- 10) Process Definition
- 11) Load Definition
- 12) Scenario Definition

To obtain a listing, the user types:

```
GENLIST [PROJECT(project)] [NOXLATE] [TERM(terminal)]
```

```
GLIST [P(project)] [N] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter specifying the project database for which a listing is desired. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[NOXLATE] is an optional parameter indicating that the listing of the model will be from a previous translation of the model. If this parameter is omitted, a translation will be performed. (The translation listing is stored in a temporary file; the user's current translation file, if there is one, is not affected by this procedure.)

[TERM(*terminal*)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

- HP - HP2647A or HP2648A terminal
- HP23 - HP2623 terminal
- TEK - TEK4105 terminal
- VT - VT100 terminal with Selanar graphics

The system responds with the following:

CURRENT PARAMETERS IN EFFECT:
VERSION: PRODUCTION VERSION 5.0
TERMINAL: Terminal type specified in command or default
PROJECT: Project specified in command or default
USER: Userid
XLATE/NOXLATE: XLATE/NOXLATE, depending upon command
ENTER YES TO PROCEED, NO TO ABORT...

A "yes" response will cause the listing to be created and a copy to be automatically printed. A "no" response will abort the command.

The listing is stored in a file named project.LST.

5.11 HARDCOPY OUTPUT OF THE PROCESS FLOWCHARTS

Hardcopy graphics of Process flowcharts are obtained in the Hardcopy User Interface (HUI). The HUI is accessed from the AISIM READY level by issuing the following command:

```
HCOPY [PROJECT(project)] [TERM(terminal)]
```

```
HC [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database with the Processes of interest. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
```

The system will respond with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL: Terminal type specified in the command or default
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the HUI to be invoked. The system will then prompt the user for all required information (see section 9 on the HUI). A "no" response will abort the command.

Note: This function is not available on VT100 or HP2648A terminals.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

5.13 INITIATING A HELP EDITOR SESSION

The Help database is accessed using commands available in the Help Editor Interface (HEI). The HEI is entered by issuing the command:

```
HEI [TERM(terminal)]  
HEI [T(terminal)]
```

where:

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP      - HP2647A or HP2648A terminal  
HP23    - HP2623 terminal  
TEK     - TEK4105 terminal  
VT      - VT100 terminal with Selanar graphics
```

The following is displayed after entering this command:

```
CURRENT PARAMETERS IN EFFECT:  
VERSION:  PRODUCTION VERSION 5.0  
TERMINAL: Terminal type specified in the command or default  
PROJECT:  Project specified in a previous command or undefined  
USER:     Userid  
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the HEI to be invoked. The HEI prompt (*) will be displayed when the system is ready to accept commands at the HEI sublevel. A "no" response will abort the command.

5.14 EXERCISING THE LIBRARY FACILITY

The Library User Interface (LUI) allows the user to do the following:

1. Move entities from a model project database into a storage area called a "buffer".
2. Move entities from a "buffer" into the database of another model project.
3. Move entities from a "buffer" into a library of model entities.
4. Move entities from a library to a "buffer".
5. Convert a version 3.0 or 4.0 project database to a version 5.0 compatible project database

The LUI is entered by issuing the command:

LIBRARY

LIB

The system will respond with the prompt:

LIBRARY READY

and the user may invoke any of the LUI sublevels listed in the LUI Command Summary (see section 10).

AISIM READY / LIST

5.15 LISTING THE CURRENT PARAMETERS

To list the current parameters in effect, type the following command:

LIST

L

The system will display the current parameters in effect, including PROJECT, USER, VERSION, and TERMINAL.

5.16 LISTING THE COMMAND PROCEDURE LINES

If a user is having problems from the AISIM READY level or LIBRARY READY level which may stem from missing system files or an operating system problem, the user can set a flag so that all of the files which control the execution of an AISIM session will be displayed as they are executed. This flag is set by typing the following command:

LISTON

When this option is in effect, all VAX/VMS commands which set up an AISIM session will be displayed at a user's terminal as they are executed. Viewing the commands as they execute may help a user determine where a problem is occurring.

ALSIM READY / LISTOFF

5.17 DISABLE THE LISTON OPTIONS

In order to disable the LISTON option, i.e., to inhibit the displaying of VAX/VMS commands as they are being executed, type the following command:

LISTOFF

This command disables the command listing mode initiated by the LISTON command.

AISIM READY / MSGOFF

5.18 DISABLE AISIM MESSAGES

Upon invoking each AISIM function, the user is presented with the current version, terminal type, project, etc., and asked if (s)he wants to continue or abort. These messages and prompt can be suppressed by typing the following command:

MSGOFF

When the user invokes a function, control will be transferred directly to that function without further prompting.

5.19 DISABLE MSGOFF FEATURE

If the user has disabled the AISIM messages and prompts via the MSGOFF command, the messages and prompts can be turned back on via the following command:

MSGON

Following this command, the user will receive the version, terminal type, project, etc. messages and prompt to continue whenever an AISIM function is invoked.

5.20 PRINTING OUTPUT REPORTS

To request printing of the model output report, type the following command:

PRINT [PROJECT(project)]

P [P(project)]

where:

PROJECT(project) is an optional parameter indicating which project's report file is to be printed. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

Result:

The output report (project.RPT) of a project is printed. This is a report of the standard results of a simulation run.

NOTE: The output report is automatically printed at the conclusion of an Analysis session.

5.21 INITIATING A REPLOT SESSION

The Replot User Interface (RUI) allows the user to display plots which were saved during previous Analysis sessions. The command to invoke the RUI is as follows:

```
REPLOT [PROJECT(project)] [TERM(terminal)]
```

```
R [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database used in creating the saved plots. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105
VT   - VT100 terminal with Selanar graphics
```

The system will respond with the following display:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 5.0
TERMINAL: Terminal type specified in command or default
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the system to complete the transfer to the RUI. The RUI prompt (\$) will be displayed when the system is ready to accept commands at the RUI sublevel. A "no" response will abort the command.

5.22 RESTORING A DATABASE (AFTER A CATASTROPHE HAS OCCURRED)

This command is used in conjunction with the BACKUP command. If the user was editing the original database and had issued a BACKUP command against this database, then a copy of the original database exists. The RESTORE command causes the damaged original database to be replaced with this backup copy.

To restore a previously backed-up database (only necessary if a catastrophe has occurred which altered the project database, or it is desirable to restart a model from a known configuration), enter the following command:

```
RESTORE [PROJECT(project)]
```

```
RESTORE [P(project)]
```

where:

[PROJECT(project)] is an optional parameter indicating the previously backed-up project database to be restored. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

The backed-up copy of the database, called project.BCK, will be copied onto the damaged database and will have the database name project.DBF.

SECTION 6

DESIGN USER INTERFACE (DUI)

The DUI and its lower levels are used to define a model by creating, modifying, or deleting AISIM model entities. The Action, Constant, Item, Load, Process, Queue, Resource, Scenario, Table, and Variable entities are created and edited at the DUI level, using the EDIT command. The descriptions of File entities can be modified using the EDIT command. The Process entities which represent operations in the modeled system are created and edited at a sublevel of the DUI level called the Process Editor Interface (PEI). The PEI is invoked by issuing the EDIT command (at the DUI level) and specifying a Process as the entity to be edited. A system architecture and its related Legal Path Table, nodes, and links are defined in a second sublevel of the DUI called the Architecture Design Editor (ADE). The ADE is invoked by issuing the ARCH command at the DUI level.

AISIM has a restricted character set for all data entered into a model. Only the characters A-Z, 0-9, "\$" (dollar sign) and "_" (underscore) may be used for model entity names and parameters. Any time the user places an invalid character in a name or a form field, the user will be requested to correct the invalid character. Any printable characters are allowed in "comment" and "description" fields of forms and in user-added help text (see section 11.2).

When creating and editing entities in the DUI level, the system prompts the user for further information by use of forms. Each form specifies the required and optional attributes of its respective entity-type. The areas on which information is to be entered appear in "reverse video" (dark characters on a light background), and indicate the attributes that are to be supplied by the user.

Each time the user presses the keyboard carriage return key, the character cursor is positioned to the start of another designated area. The user enters parameters requested by the form by keying in the desired alphanumeric information. If the user changes his mind about the parameters previously keyed in, he may alter them by merely writing over the old information. When the user is satisfied with the contents of the form, he inputs it to the computer by exiting the form. Below is a complete description of the use of forms.

While the user is in the DUI, all changes are made to a working copy of the user's database. When the user issues a SAVE command during or at the end of the DUI session, the working database is copied back into the user's real database. This procedure enables the user to change his/her mind about changes made in the working database and to protect the user's real database in case the computer crashes during a DUI session.

The AISIM DUI commands used to input, modify, and delete entities from the model, are illustrated in figure 6-2 and described on the pages that follow it.

USE OF THE FORMS EDITOR

This section describes the use of the forms editor on the various terminals. Figure 6-1 is a chart which describes the keys used to achieve specific movements through a form. Following the figure is a description of each of the ways of moving through a form.

	UP	DOWN	LEFT	RIGHT	ENTER	+FIELD	-FIELD
HP2647A	F1	F2	F3	F4	F5	<cr>	F6
HP2648A	F1	F2	F3	F4	F5	<cr>	F6
HP2623	F1	F2	F3	F4	F5	<cr>	F6
TEK4105	F1	F2	F3	F4	F5	<cr>	F6
VT100	↑	↓	<--	-->	PF1	<cr>	PF2

Figure 6-1. Terminal Profiles

UP - If the cursor is in a block of fields, such as Resource attributes, the cursor will move up to the field above it. If the cursor is in a single field or at the top of a block, the cursor will move to the end of the next field above it. If there are no fields above it, the cursor will wrap to the end of the last field in the form.

DOWN - If the cursor is in a block of fields, such as Resource attributes, the cursor will move down to the field below it. If the cursor is in a single field or at the bottom of a block, the cursor will move to the beginning of the next field below it. If there are no fields below it, the cursor will wrap to the beginning of the first field in the form.

LEFT - The cursor will move one position to the left in the current field. If the cursor is at the beginning of a field, it will move to the end of the previous field. If the cursor is at the top of the form, it will wrap to the end of the last field in the form.

RIGHT - The cursor will move one position to the right in the current field. If the cursor is at the end of a field, it will move to the beginning of the next field. If the cursor is at the end of the form, it will wrap to the beginning of the first field in the form.

ENTER - Exit the form and send the data in the form to be processed by the AISIM system.

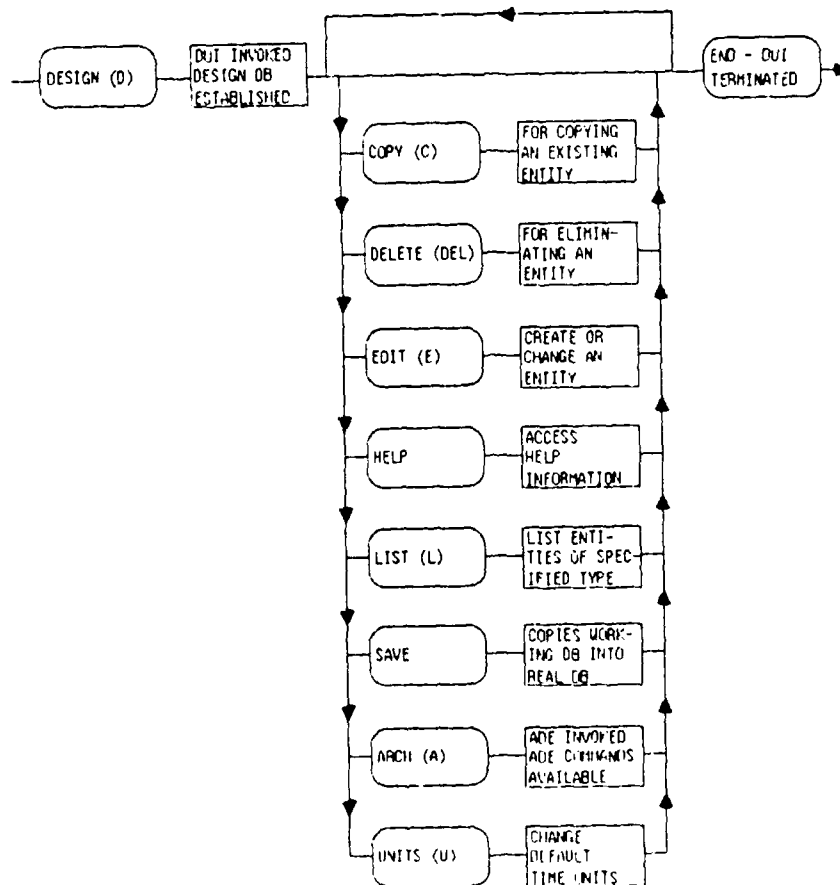


Figure 6-2. Design User Interface Commands

6.1 DUI COMMAND SUMMARY

Figure 6-3 contains a summary of the DUI level commands.

ARCH
A

COPY {entity-type},{existing-name},{new-name}
C

DELETE {entity-type},{entity-name}/*
DEL

EDIT {entity-type},{entity-name},{OLD/NEW}
E

END

HELP {subtopic},...,{subtopic}
HELP [@topic],[topic-name],[subtopic],...,{subtopic}

LIST {entity-type}
L

SAVE

UNITS {units-type}
U

Figure 6-3. DUI Command Summary

6.1.1 DUI COMMAND: ARCH

The ARCH command is used to invoke the Architecture Design Editor (ADE).

This command is valid only in the DUI Ready Level.

COMMAND SYNTAX:

ARCH

A

FUNCTION RESULT:

The ADE is invoked so that the architecture is built under the project designated by the DESIGN command. A "#" prompt is provided for the user to input ADE commands. These commands are discussed in section 6.3.

6.1.2 DUI COMMAND: COPY

The COPY command is used to create a copy of an existing entity.

COMMAND SYNTAX:

```
COPY {entity-type},{existing-name},{new-name}
```

```
C
```

where:

{entity-type} is a required parameter indicating any valid entity type.

Entity-type may be any of the following:

Entity-type	Acceptable Abbreviation
Action	A
Constant	C
Item	I
Load	L
Process	P
Queue	Q
Resource	R
Scenario	S
Table	T
Variable	V

{existing-name} is a required parameter identifying the existing entity whose parameters are to be duplicated.

{new-name} is a required parameter which specifies the name of the new entity whose parameters are duplicates of the "existing entity".

If entity type, existing-name or new-name is missing or invalid, the user is prompted.

A carriage return entered in response to any prompt aborts the command and returns the user to the DUI Ready state - * prompt.

6.1.3 DUI COMMAND: DELETE

The DELETE command is used to eliminate a named entity of a given type from the user database. A restriction on the use of this command is that Resources associated with architectural nodes or links cannot be deleted outside of the Architecture Design Editor sublevel.

COMMAND SYNTAX:

```
DELETE {entity-type},{entity-name}
      {entity-type},{entity-name},...,{entity-name}
      {entity-type},*
DEL
```

where:

{entity-type} is a required parameter indicating any valid entity type. The valid entity types are listed in section 6.1.2.

{entity-name} is a required parameter indicating the name of the entity to be deleted. It is permissible to give a list of entity-names, of the same type, each member of which is separated by a comma.

* is a parameter used indicate all of the entities of the specified type are to be deleted.

If entity-type or entity-name is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the DUI Ready state - * prompt.

FUNCTION RESULT:

If the named entity is not a Resource associated with a architectural node or link, the entity will be deleted from the user's working database. If the entity is a Resource associated with a node or link, the user will be given the message:

" entity " IS ASSOCIATED WITH THE ARCH. AND CAN ONLY BE DELETED IN THE ADE

where " entity " is the name of the entity to have been deleted.

When there is more than one such Resource listed in the command to delete the user will be given the above message for each one.

6.1.4 DUI COMMAND: EDIT

The EDIT command is used either to create an entity, or to change an existing entity.

COMMAND SYNTAX:

```
EDIT {entity-type},{entity-name},{OLD/NEW}
```

E

where:

{entity-type} is a required parameter indicating any valid entity type. The valid entity types are listed in section 6.1.2 and additionally include "File".

{entity-name} is a required parameter indicating the name of the entity to be edited.

[OLD/NEW] is an optional parameter indicating that the named entity is to be created (NEW), or that the named entity exists (OLD) and is to be changed. If the [OLD/NEW] parameter is entered incorrectly, the user is prompted for confirmation to continue the command. The default for this parameter is OLD.

FUNCTION RESULT:

If the entity-type specified is Process, the PEI level (see section 6.2) is automatically invoked. If any other valid entity type is specified, the user is presented with a form to describe that entity. The forms for the entities are shown in figures 3-1 through 3-4, 3-6 through 3-8, 3-42, and 3-44. The user must fill out the form to input the completed entity into the working database. The user is then returned to the DUI Ready state - *

prompt.

DUI / END

6.1.5 DUI COMMAND: END

The END command is used to terminate a DUI session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The Design session is ended. The working database is closed. If a SAVE command has not been given since the last EDIT command, the user is asked if the working database is to be saved. The query is:

SAVE (Y/N)?

If the user answers "Y", the working database is saved into the real database and the session is ended. Control is passed to the AISIM READY level (level 3). If the user answers "N", the session is ended and the working database is not saved. Control is passed to the AISIM READY level (level 3). Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready level - * prompt.

6.1.6 DUI COMMAND: HELP

The **HELP** command provides the user with access to help information about the current DUI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the DUI function is displayed. The commands acceptable at the current DUI level of operation are listed as subtopics indicating that further help is available on them. **HELP** with no path specified displays the DUI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a **HELP** message on the subtopic will be displayed. If one or more subtopics exist at this level, **HELP** will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

6.1.7 DUI COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name and description.

COMMAND SYNTAX:

```
LIST {entity-type}
```

```
L
```

where:

{entity-type} is a required parameter indicating any valid entity type. The valid entity types are listed in section 6.1.2 and additionally include "File".

If {entity-type} is missing or invalid, the user is prompted for a valid entity type.

A carriage return entered in response to the prompt aborts the command, and the user is returned to the DUI Ready state - * prompt.

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type.

6.1.8 DUI COMMAND: SAVE

The SAVE command copies the contents of the working database into the user's permanent database.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The real database is replaced with the contents of the working database, and the user is returned to the DUI ready state - * prompt. The command is useful when the user is defining a large system. With the SAVE command the user saves the model design up to the point at which the command is given. This protects that portion of the design from computer failures.

6.1.9 DUI COMMAND : UNITS

The UNITS command is used to set the default time units that will appear in the forms for Load and Scenario entities and Action Primitives created after this command is issued.

COMMAND SYNTAX:

UNITS {units-type}

U

where:

{units-type} is a required parameter indicating the units to be used. The valid units and unit abbreviations and their meanings are as follows:

<u>Command entry</u>		<u>meaning</u>
nseconds (ns)	-	nanoseconds
useconds (us)	-	microseconds
mseconds (ms)	-	milliseconds
seconds (s)	-	seconds
minutes (m)	-	minutes
hours (h)	-	hours
days (d)	-	days

FUNCTION RESULT:

The default time units used in Load, Scenario, and Action Primitive forms will be set to the entered value.

6.1.10 Termination of a DUI Session

As mentioned earlier, a DUI session is terminated by issuing the END command. Syntax and results are described in the preceding section. The DUI session is ended. The working database is closed. If a SAVE command has not been given since the last EDIT command, the user is asked if the working database is to be saved. The query is:

SAVE (Y/N)?

If the user answers "Y", the working database is saved into the real database and the session is ended. If the user answers "N", the session is ended and the working database is not saved. Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready state. When the SAVE query is answered, control is returned to the AISIM READY level and the AISIM READY prompt is displayed.

There are two modes in the PEI: DRAW and NODRAW. Under DRAW mode, all changes to Primitives on the screen are reflected in the display. Under NODRAW mode, changes are not reflected in the display until the user explicitly requests that the display be updated. When the user first enters the PEI, DRAW mode is the default. If the user changes the mode, the change will stay in effect for all subsequent uses of the PEI until changed by the user or the user exits the DUI. These modes are explained more fully in the PEI DRAW and NODRAW commands (sections 6.2.6 and 6.2.11).

BOTTOM B

CHANGE {position}
C

DELETE {first position},{number of consecutive positions}
DEL

DOWN {number of positions}
D

DRAW
DR

END
E

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

HOLD {position}
H

MENU
M

NODRAW
N

PLACE {Primitive},{position}
P

REDRAW
RED

Figure 6-4. PEI Command Summary

TOP
I

UP [number of positions]
U

Figure 6-4. PEI Command Summary (cont'd)

PEI / BOTTOM

6.2.2 PEI COMMAND: BOTTOM

The BOTTOM command is used to display the last six Primitives in the current Process structure.

COMMAND SYNTAX:

BOTTOM

B

FUNCTION RESULT:

The bottom of the Process structure being edited is drawn from the END symbol up. The END symbol is always the last position of a Process structure.

6.2.3 PEI COMMAND: CHANGE

The CHANGE command is used to modify the user defined parameters of a Primitive within the current Process structure.

COMMAND SYNTAX:

CHANGE {position}

C

where:

{position} is a required parameter indicating the position of the Primitive, within the Process structure, whose parameters are to be changed.

FUNCTION RESULT:

When the CHANGE command is invoked, the user is presented with a form corresponding to the Primitive at the indicated position. The user may change any or none of the attributes of the Primitive. If the user is in DRAW mode, the Process structure is then redisplayed with any changes made; otherwise, the screen remains unchanged. If the changed Primitive was not on the screen, it is redrawn at screen position three. If the user is in NODRAW mode, the top of the display is set to be two above the changed Primitive, but the Process is not redrawn.

6.2.4 PEI COMMAND: DELETE

The DELETE command allows the user to delete a single Primitive, or a range of Primitives, from the current Process structure.

COMMAND SYNTAX:

DELETE {first position},{number of consecutive positions}

DEL

where:

{first position} is a required parameter indicating the position of the first Primitive to be deleted.

{number of consecutive positions} is an optional parameter indicating the number of consecutive positions to be deleted, starting with the Primitive indicated by the {first position} parameter. If this parameter is omitted, the default condition is to delete only the Primitive at the position indicated by the {first position} parameter.

FUNCTION RESULT:

The Primitives indicated by the {first position} parameter and the optional parameter are deleted from the Process structure. The START and the END symbols may not be deleted. Additionally, the numbers of all Primitives being deleted must be displayed on the screen.

If the user is in DRAW mode, this simply means that the Primitives to be deleted must be visible. If the user specifies to delete Primitives past the end of the screen, only the Primitives on screen will be deleted. After the delete command is issued, the remaining Primitives in the structure are scrolled up.

If the user is in NODRAW mode, the numbers of the primitives being deleted must be on screen, but not necessarily the symbols themselves. For example, say the first six Primitives are being displayed and the user deletes Primitives three through six. Since the legend still shows three through six, the user can delete the new third through sixth Primitives even though the symbols on screen may not correspond to the Primitives being deleted. The user should take care when deleting Primitives while in NODRAW mode. If the user specifies to delete Primitives whose numbers are past the end of the screen, only the Primitives whose numbers are on screen will be deleted.

6.2.5 PEI COMMAND: DOWN

The DOWN command allows the user to "jump down" the current Process structure an indicated number of positions.

COMMAND SYNTAX:

DOWN [number of positions]

D

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to "jump down". If this parameter is not used, the default condition is to drop the Process structure down six Primitives, which is analogous to displaying the next page.

FUNCTION RESULT:

The Process structure jumps down the number of positions indicated by the optional parameter, if given. Otherwise the structure jumps down six Primitives or to the bottom of the structure if less than six Primitives follow the last position currently displayed.

6.2.6 PEI COMMAND: DRAW

The DRAW command is used to put the user in the PEI DRAW mode.

COMMAND SYNTAX:

DRAW

DR

FUNCTION RESULT:

The DRAW command sets the PEI mode to DRAW mode. This mode will remain in effect for all future PEI sessions during the current DUI session until changed by a NODRAW command.

In DRAW mode, all changes made by a user will be reflected in the display. I.e., if a Primitive is changed via the CHANGE command, that Primitive will be redrawn on the screen. If Primitives on the screen are deleted, remaining Primitives will be scrolled up to fill the display.

PEI / END

6.2.7 PEI COMMAND: END

The END command is used to terminate and exit the PEI session.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The PEI session is ended, the graphics display is erased, and the user is returned to the DUI Level.

6.2.8 PEI COMMAND: HELP

The HELP command provides the user with access to help information about the current PEI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the PEI function is displayed. The commands acceptable at the current PEI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the PEI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

6.2.9 PEI COMMAND: HOLD

The HOLD command allows the user to insert any valid Primitive, which is already a part of the current Process structure, into the menu item "HOLD" so that it may be replicated.

COMMAND SYNTAX:

HOLD {position}

H

where:

{position} is a required parameter indicating the position of the Primitive which is to be placed in hold for the purpose of replication.

FUNCTION RESULT:

The Primitive (complete with the previously defined parameters) is placed in HOLD. This item may then be replicated by using the PLACE command and using HOLD as the Primitive to be placed. When a Primitive is stored in HOLD, it remains there, accessible to the user throughout the DUI session, and thus Primitives may be moved from one Process to another. When there is a Primitive in HOLD on a terminal on which the menu can be displayed (see MENU command), the name of the Primitive being held appears below the menu display area preceded by an asterisk (example: *CREATE).

6.2.10 PEI COMMAND: MENU

MENU is used to display the possible Primitives for a Process.

COMMAND SYNTAX:

MENU

M

FUNCTION RESULT:

The menu is a one-column list of names of the valid Primitives (see section 3.9 for a description of the Primitives). If the menu will fit on the screen, it is displayed to the left of the Process flowchart. If the menu will not fit on the screen, a message will be displayed noting that fact. The menu can be displayed on HP2647A, HP2648A, and TEK4105 terminals. Figure 6-5 shows the Process menu.

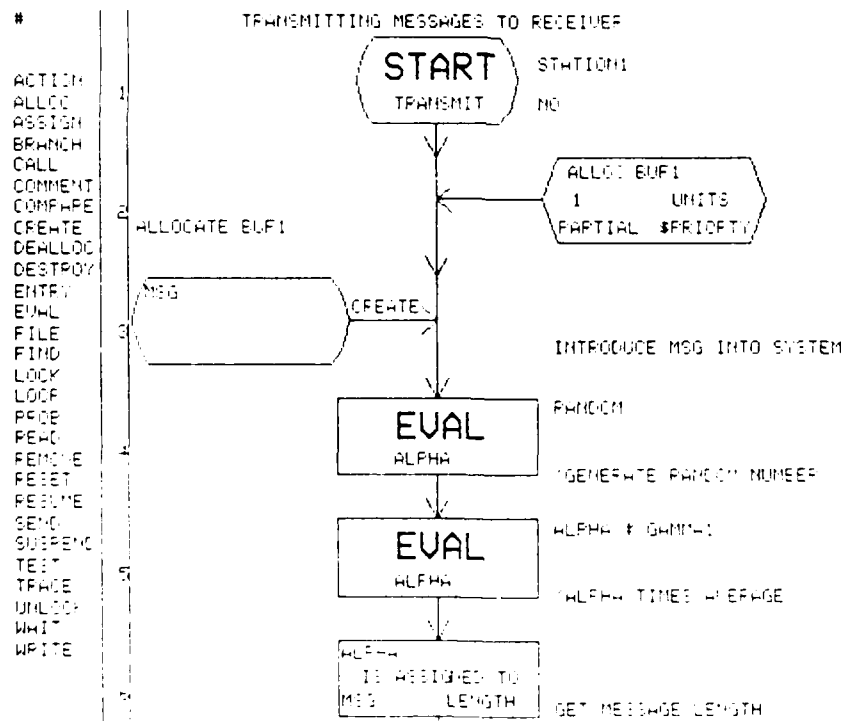


Figure 6-5. Process Display with Menu

6.2.11 PEI COMMAND: NODRAW

The NODRAW command is used to put the user in the PEI NODRAW mode.

COMMAND SYNTAX:

NODRAW

N

FUNCTION RESULT:

The NODRAW command sets the PEI mode to NODRAW mode. This mode will remain in effect for all future PEI sessions during the current DUI session until changed by a DRAW command.

In NODRAW mode, no changes which are made to Primitives in the Process are reflected in the display until the display is explicitly redrawn by the user. Commands which can be used to update the display are TOP, BOTTOM, UP, DOWN and REDRAW. The user should take care when deleting Primitives while in NODRAW mode to guard against deleting necessary Primitives since the screen is not updated after a DELETE is performed.

6.2.13 PEI COMMAND: REDRAW

The REDRAW command is used to update the current Process display. This command is generally used when the user is in NODRAW mode.

COMMAND SYNTAX:

REDRAW

RED

FUNCTION RESULT:

This command causes the Process display to be redrawn from the location at which the last change was made. For example, if the last change was made at the top of the Process, the display will be drawn from the top of the Process. Other portions of the Process can be displayed using the TOP, BOTTOM, UP and DOWN commands. The REDRAW command is especially useful when the user is deleting Primitives in NODRAW mode so the user can see what the Process really looks like.

6.2.14 PEI COMMAND: TOP

The TOP command is used to display the first six Primitives in the current Process structure.

COMMAND SYNTAX:

TOP

T

FUNCTION RESULT:

The first six Primitives of the Process structure being edited (or the entire Process if the structure consists of no more than six Primitives) are drawn from the START symbol down. The START symbol is always the first position of a Process structure.

6.2.15 PEI COMMAND: UP

The UP command allows the user to "jump up" the current Process structure an indicated number of positions.

COMMAND SYNTAX:

UP [number of positions]

U

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to "jump up". If this parameter is not used, the default condition is to "jump up" the Process structure six Primitives, which is analogous to displaying the previous page.

FUNCTION RESULT:

The Process structure jumps up the number of positions indicated by the optional parameter, if given. Otherwise the structure jumps up six Primitives or to the top of the structure if less than six Primitives precede the first position currently displayed.

6.2.16 Terminating a PEI Session

Only one Process can be created or edited during a PEI session. To create or edit other Processes or change to another level the user must terminate the current PEI session and return to the DUI level. This is accomplished by giving the END command described in section 6.2.7. The current working database is left open and control is transferred to the DUI level.

6.3 ARCHITECTURE DESIGN EDITOR (ADE)

The ADE is used to define the layout and interconnection of the physical aspect of a data processing network. It is not necessary to develop an architecture model if the user wishes to model operations without regard to where these operations take place. However, if Items are routed through a system or if Processes at one location trigger Processes in another, then an architecture model is necessary.

The ADE allows the user to create graphically a picture of the system architecture by positioning symbols and connections. It also allows the user to define the legal paths of communication between the connections (and along the connections).

Even if a user has defined a Legal Path Table while creating an architecture, the system offers the option of automatically building a Legal Path Table. The user is queried to resolve any ambiguities. The Legal Path Table is used during the simulation to control the routing of Items that are being passed through the system.

It is important to note that each node and link represented in the architecture is intended to represent some system resource such as a CPU, disk drive, tape drive, or channel. The system automatically creates model Resources for these system Resources. The parameters of such Resources can be altered both in the ADE--though the DEFINE command (see section 6.3.6)--and in the DUI--with the EDIT command (see section 6.1.4).

Hardcopies of a created architecture can be reproduced using a graphics device (see appendix A.4).

6.3.1 Concepts for Using ADE

This section is intended to familiarize the user with the capabilities of the ADE so that he may better understand the description of its use in sections further below.

The view space is divided by vertical and horizontal grids. Grid lines running vertically mark off the position and are numbered starting with zero at the left side. Grids running horizontally mark off the Y position and are identified with numbers, starting with zero at the bottom. Another aid to building the architecture is variable symbol size. The user can specify the size of symbols as he positions them in the view space. The user is provided with commands to change his view screen position, to position nodes which represent system Resources, to delete nodes, and to change symbol names and sizes. A command is provided which allows the user to specify connections between nodes. These connections (or links) are defined as system Resources. Any two nodes may be connected by more than one link, but there may be only one legal path between these two nodes. (Exception: When using Method A, B, or C algorithms to define the Legal Path Table, two node types "TTY" and "LOD" are considered "leaf-nodes" and should have only one connection to one other node. The architecture developed using the ADE becomes the basis

for generating the Legal Path Table which is used to route Items through a system.

The view screen on the HP2647A terminal, for example, is approximately five inches high by eight and one-half inches wide. This workspace is too small for some systems. The ADE, therefore, gives the user a workspace which is thirteen and two-tenths inches high by 20 inches wide and allows the user to move the viewspace anywhere in this workspace to construct the architecture. The contrast between viewspace and workspace is illustrated in figure 6-6. The workspace is the same size on all terminals supported by AISIM.

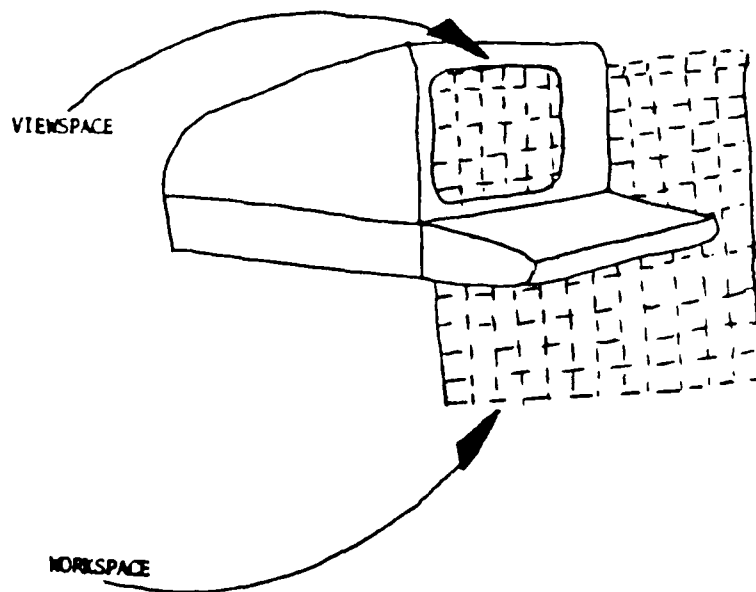


Figure 6-6. Viewspace versus Workspace in ADE

6.3.2 Use of the ADE

The ADE can only be accessed from the DUI level. The ADE level is entered by issuing the following command:

ARCH

A

Only one architecture is allowed per design database. This prevents the user from specifying an architecture structure that does not relate to the Processes and Resources that have been defined. Experiments using common Processes, Resources, etc. with different architectures can be run by following the procedure listed below:

- 1) While in the VAX/VMS ready level or AISIM READY level, COPY the project.DBF data file to newproject.DBF data file where: project and newproject are names of PROJECT databases for AISIM models.
- 2) Enter the ADE to edit the architecture contained in newproject.DBF.
- 3) Simulations can now be run using the newproject database.

If there is no architecture defined in the design database, the system will provide a blank grid on the screen and a pound sign (#) prompt for the user to enter commands. If an architecture has already been defined, then the old architecture will be displayed and the user will be provided a pound sign (#) prompt for entering commands.

The ADE has DRAW and NODRAW modes which are similar to the PEI DRAW and NODRAW modes. However, if a user is logged on to a VT100 terminal, only NODRAW is available. In NODRAW mode, the user can place and change symbols, connect nodes, and perform all of the functions of the ADE, except that the results of the commands will not be reflected in the screen until the user explicitly redraws the screen with a REDRAW or WINDOW command. If the user is in DRAW mode on a supported terminal, the results of all ADE commands will be reflected in the display. The VT100 is always in NODRAW mode. The default for the other terminals is DRAW mode.

The following pages give a summary of commands available in the ADE and their use. These commands are legal in the ADE level only.

```

CHANGE NAME,{name},{new-name}
        TYPE,{name},{type}
        SIZE,{name},{size}

CHG

CONNECT {node1},{node2},{link}[_F]
CON

DEFINE {symbol-type},{Resource-name}
        PATH,{node1},{node2},{link1},...,{linkn} DEF

DELETE {name1},...,{name-n}
        PATH,{node1},{node2}
        *

DEL

DRAW
DR

END

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST PATH,{node1},{node2}
        LPT

L

MOVE {node},{x-position},{y-position}
M

NODRAW
N

PLACE {symbol-type},{node},{x-position},{y-position},{size}
P

RECON {link}
R

REDRAW
RED

SAVE

WINDOW {direction1},{n},{direction2},{n}
W

```

Figure 6-7. ADE Command Summary

ADE SYMBOLS

6.3.3 ADE Symbols

Symbols used to construct an architecture are generic in nature. The shape associated with some symbols is representative of a computer system's hardware elements although no implicit attributes of computer hardware elements are given to the symbols. Attributes defined for a symbol which make it represent an actual physical device must be defined by the user. Attributes are attached to symbols by the **DEFINE** command.

Symbols in an architecture correspond directly with Resources. This relationship applies to nodes and links. All symbols which are directly connected correspond to an entry in the Legal Path Table.

One other implied relationship applies to the symbols in an architecture. The symbols **TTY** and **LOD** are considered to be "terminal" symbols by the Legal Path Table. Therefore, these two symbols have a constraint that they can be connected with only one link to one of the other symbol types. Also, **TTY** and **LOD** symbols cannot be directly connected. These constraints are enforced by the LPT generation not the ADE.

The complete symbol set for AISIM architecture is shown in figure 6-8.

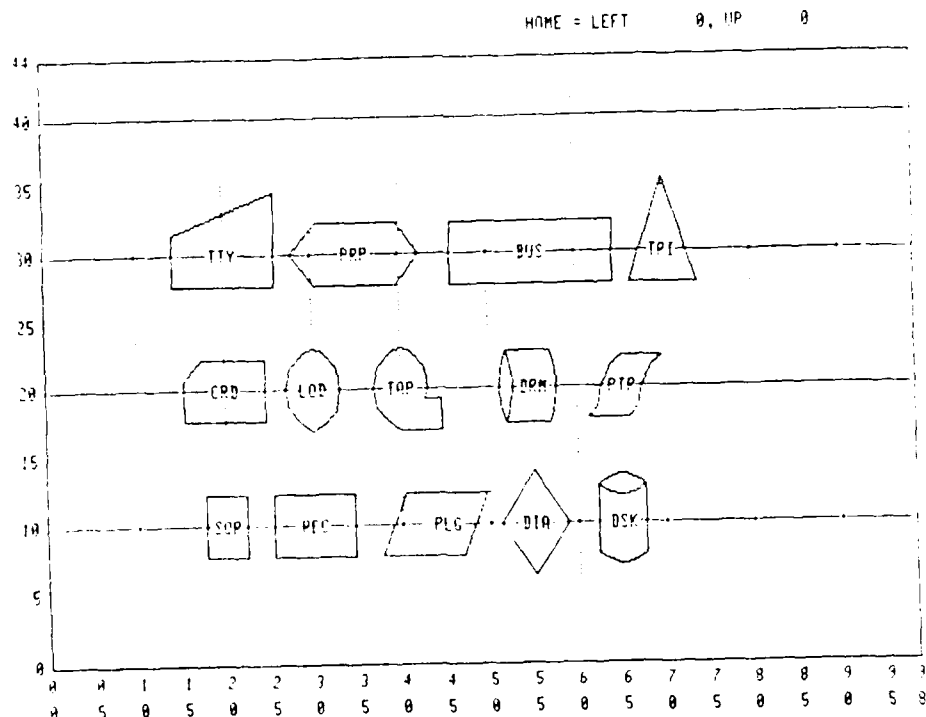


Figure 6-8. Architecture Symbols

6.3.4 ADE COMMAND: CHANGE

The CHANGE command allows the user to modify the name, type, or size of an ADE symbol which represents an architecture node.

COMMAND SYNTAX:

CHANGE NAME,{name},{new-name}

CHANGE TYPE,{name},{type}

CHANGE SIZE,{name},{size}

CHG

where:

{name} is a required parameter indicating the name of the symbol which is to be changed. For the commands CHANGE TYPE and CHANGE SIZE, name must designate a node.

{new-name} is a required parameter specifying a new name for the current named symbol where new name should be 1-8 alphanumeric characters.

{type} is a parameter specifying that the named symbol is to be changed from its current type to "type" which is one of the legal symbol types. The symbol types are shown in figure 6-8.

{size} is a required parameter specifying that the named symbol is to be changed from its current size to "size" where size can be 1-20.

FUNCTION RESULT:

The indicated changes are made to the symbol "name". When the user changes a symbol type or size, there is no impact on the other parameters. When the name is changed, the default size is the number of characters in the name. If the user is in DRAW mode, the symbol is redrawn to reflect the changes.

6.3.5 ADE COMMAND: CONNECT

The CONNECT command is used to show connections between architecture nodes by placing links between them.

COMMAND SYNTAX:

```
CONNECT {node1},{node2},{link}[_F]
```

```
CON
```

where:

{node1} is a required parameter indicating the first symbol of a from-to pair of symbols to be connected and where node1 is 1 to 8 alphanumeric characters.

{node2} is a required parameter indicating the second symbol of a from-to pair of symbols which are to be connected and where node2 is 1 to 8 alphanumeric characters.

{link} is a required parameter indicating the name of the connection which is to be made and where link is 1 to 8 alphanumeric characters.

[_F] is an optional parameter appended directly to link indicating that the communication link between nodes node1 and node2 is full-duplex. The name of the link must be no longer than eight characters including the "F". The effect of this is to create two links, a "link_A" and a "link_B". Links defined without this parameter bear a half-duplex default.

FUNCTION RESULT:

If node1 is not in the viewspace when the command is issued, the user will be prompted with the message,

THE FROM NODE MUST BE ON THE SCREEN TO ESTABLISH CONNECT: COMMAND ABORTED:

If node1 is on the viewspace and the user is on a terminal other than a VT100, a cursor (+) is turned on. If the user is on an HP terminal, the cursor appears at node1. If the user is on a TEK4105 terminal, the cursor appears where it was last positioned, or at the lower left corner if it was never moved. At this point, the user has two alternatives:

- 1) he may cause the system to connect the two symbols with a straight line through their centers by depressing any non-period, alphanumeric character or,
- 2) he may cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:

- a) moving the cursor using the graphics controls, to a position where he wishes to bend the line,
- b) typing a period (.),
- c) repeating a) and b) until a maximum of five corners have been created.
- d) completing the line segment from the last corner to symbol 2 by entering a non-period alphanumeric character.

Alternative 2 allows the user to place symbols randomly and later show connections that would be obscured or confusing if generated by Alternative 1. Connections can be straightened or have corners added to them with the RECON command (see section 6.3.15).

If the user is on a VT100 terminal, the two nodes are automatically connected by a straight line. Bent line connections are not possible.

If the user is in NODRAW mode on a terminal other than a VT100, the connect command operates as stated above for DRAW mode except that the line or line segments are not reflected on the screen. Thus the user can still make connections while in NODRAW mode.

After a connection is defined, two entries are entered in the Legal Path Table. The first is an entry for the path from node1 to node2 via link, and the second entry specifies a path from node2 to node1 via link. If link is defined as full-duplex, then the path from node1 to node2 uses "link_A", while the path from node2 to node1 uses "link_B". (See section on "Define" command). Node1 is then established as the link's from node and node2 is established as the link's to node. All subsequent paths using this full-duplex link will use "link_A" if they go in the direction of the from node to the to node and will will use "link_B" if they go in the opposite direction.

6.3.6 ADE COMMAND: DEFINE

The DEFINE command serves two functions. It is used to define attributes to be associated with symbols (this allows the user to make the logical assignment of physical device characteristics to the Resource). DEFINE is also used to indicate the legal path between nodes in the architecture.

COMMAND SYNTAX:

```
DEFINE {symbol-type},[Resource-name]
```

```
DEFINE PATH,{node1},{node2},{link1},...,{linkn}
```

```
DEF PATH
```

where:

{symbol-type} is the symbol type (sqr,dia,lod,TTY,etc.) for which the user wishes to define attributes. Figure 6-8 shows these symbols.

[Resource-name] is an optional parameter that specifies the name of an existing Resource from which the symbol-type attributes are to be copied.

{node1} is the name of the node from which the path is to run.

{node2} is the name of the node to which the path is to run.

{link1},...,{linkn} are the names of the links along which the legal path between node1 and node2 is to run.

FUNCTION RESULT:

If the DEFINE command is issued with the format

```
DEFINE {symbol-type}
```

a form will be displayed that shows the parameters currently assigned to this symbol type. The form has the same format as the Resource form in figure 3-6. The user may modify these parameters as desired. After symbol attributes have been defined, any further Resources automatically created in association with the symbol will be given the attributes that were defined for that symbol type.

If the syntax of the command is:

```
DEFINE {symbol-type},[Resource-name]
```

the system will present the user with a form to be filled with the attributes of the named Resource. The user can check the data and/or modify it. When entered, the data last displayed in the form will be used to create the attributes of the symbol type.

- c) Any path which uses the link CON_F in the direction from its to node to its from node will use CON_B.
- d) Establishing the connection between two nodes implicitly defines a point-to-point path between them.

These four rules have a number of restrictions of which the user should be aware:

- 1) Defining a path from one node to another implies defining paths from all nodes along the path to the last node in the path.
- 2) Changing a path (redefining, deleting) changes any other paths that use it as a sub-path.
- 3) A point-to-point path cannot be deleted.
- 4) When a path between two directly connected nodes is deleted, a point-to-point path is automatically restored.
- 5) Deleting a node or link from an architecture removes any paths which use the deleted entity.
- 6) Changing the name of a node or link changes the name of the entities in the Legal Path Table as well.
- 7) Cyclic paths are not allowed.

6.3.7 ADE COMMAND: DELETE

The DELETE command allows the user to delete nodes or links in the architecture or parts (or all) of the previously defined Legal Path Table (LPT).

COMMAND SYNTAX:

DELETE {name1},...,{name-n}

DELETE PATH,{node1},{node2}

DELETE *

DEL

where:

{name1} is a required parameter that specifies the node or link to be deleted.

{name-n} is an optional parameter which specifies an additional node or link to be deleted.

{node1} and {node2} are required parameters indicating the nodes between which the legal path is to be deleted.

* indicates the entire architecture is to be deleted.

FUNCTION RESULT:

If the user is in DRAW mode, the following results are seen. When a symbol is being deleted, the symbol and all connections to it are erased from the screen and removed from the database. If a connection is being deleted, the connection is erased from the screen and is removed from the database.

If the user is NODRAW mode, the affected entries are deleted from the database and the screen remains unchanged. When a path between node1 and node2 is deleted from the Legal Path Table, only that path is deleted; any sub-paths which are in this path are unaffected.

6.3.8 ADE COMMAND: DRAW

The DRAW command is used to put the user in the ADE DRAW mode.

COMMAND SYNTAX:

DRAW

DR

FUNCTION RESULT:

The DRAW command sets the ADE mode to DRAW mode. This mode will remain in effect for all future ADE sessions during the current DUI session until changed by a NODRAW command. The DRAW command is not available on a VT100 terminal.

In DRAW mode, all changes made by a user to the architecture which affect the architecture display are immediately reflected in the display, i.e., all nodes and connections are drawn on the screen as they are added to the architecture, and deleted nodes and connections are erased from the architecture.

6.3.9 ADE COMMAND: END

The END command is used to terminate the ADE session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The END command terminates the edit mode of the ADE session and automatically triggers the generation of a Legal Path Table (LPT). The user will be questioned as to the method of generation for the LPT and for information necessary to clear up ambiguities in its generation before control is returned to the DUI level. The LPT is described in section 6.3.19.

If the user does not wish to generate an LPT, another END command will return control to the DUI level.

6.3.10 ADE COMMAND: HELP

The HELP command provides the user with access to help information about the current ADE interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified help information on the ADE function is displayed. The commands acceptable at the current ADE level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the ADE help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

6.3.11 ADE COMMAND: LIST

The LIST command enables the user to list the legal paths that have been defined in the architecture.

COMMAND SYNTAX:

LIST PATH,{node1},{node2}

LIST LPT

L

where:

{node1} is the name of the node at which the path to be listed begins.

{node2} is the name of the node at which the path is to end.

FUNCTIONAL RESULT:

If the command syntax is LIST PATH, a format like that below is displayed:

FROM: node3 TO: node2 PATH:

link1,link2,...,linkn

If the command syntax is LIST LPT, the entire Legal Path Table is displayed.

6.3.12 ADE COMMAND: MOVE

The MOVE command allows the user to change the location of a node in the architecture.

COMMAND SYNTAX:

```
MOVE {node},{x-position},{y-position}
```

M

where:

{node} is the name of the node to be moved.

{x-position} is the x-coordinate of the new position, i.e., the position to which the node is to be moved.

{y-position} is the y-coordinate of the new position, i.e., the position to which it is to be moved.

FUNCTION RESULT:

If the user is in DRAW mode, the node and all links to or from it will first disappear from the screen. The node will then be redrawn at the new position and the previously defined connections with other nodes will reappear.

If the user is in NODRAW mode, the coordinates of the node will be changed in the database, and the screen will remain unchanged.

6.3.13 ADE COMMAND: NODRAW

The NODRAW command is used to put the user in the ADE NODRAW mode.

COMMAND SYNTAX:

NODRAW

N

FUNCTION RESULT:

The NODRAW command sets the ADE mode to NODRAW mode. This mode will remain in effect for all future ADE sessions during the current DUI session until changed by a DRAW command (section 6.3.8).

In NODRAW mode, no changes which are made to the architecture are reflected in the display until the display is explicitly redrawn by the user. For example, when nodes are placed in the architecture or deleted from the architecture, the changes are made to the database, but the screen remains unchanged. Commands which can be used to update the display are REDRAW (section 6.3.16) and WINDOW (section 6.3.18) commands.

All ADE commands are available while in NODRAW mode, except that on a VT100 terminal, connections can only be straight lines - bent line connections are not allowed. Connections on the VT100 are drawn automatically when the CONNECT (section 6.3.5) and RECON (section 6.3.15) commands are issued.

6.3.14 ADE COMMAND: PLACE

The PLACE command allows the user to position a legal ADE symbol in the view space at specified coordinates.

COMMAND SYNTAX:

```
PLACE {symbol-type},{node},{x-position},{y-position},{size}
```

P

where:

{symbol-type} is a required parameter which specifies one of the legal ADE symbol types. The legal symbol types are shown in figure 6-8.

{node} is a required parameter that indicates the name that is to be displayed and associated with this placement of a symbol and where name is 1 to 8 alphanumeric characters.

{x-position} is a required parameter that specifies the horizontal position of the symbol relative to vertical grid number position 0. The x-position must be within the limits of the view screen.

{y-position} is a required parameter that specifies the vertical position of the symbol relative to horizontal grid position 0. The y-position must be within the limits of the view screen.

[size] is an optional parameter specifying the size of the symbol to be placed. The default size is the number of characters in name. Legal sizes are 1-20.

FUNCTION RESULT:

If the user is in DRAW mode, a symbol of the specified type appears on the view screen at the x, y positions indicated in the command. The symbol name appears within the symbol and the symbol size is regulated by the size parameter.

If the user is in NODRAW mode, the symbol is added to the database, and the screen remains unchanged.

6.3.15 ADE COMMAND: RECON

The RECON command allows the user to alter the shape of a given link, giving it corners, decreasing the number of corners it has, or adding to the number of corners it has.

COMMAND SYNTAX:

RECON {link}

R

where:

{link} is the name of the link to be redrawn.

FUNCTION RESULT:

If the user is in DRAW mode, the link will disappear, but the cursor (+) will be turned on. The cursor is positioned at the from node on an HP terminal, or at its last location on a TEK4105 terminal (see CONNECT command). As with the CONNECT command (section 6.3.5), the user has two alternatives:

- 1) cause the system to connect the two symbols with a straight line through their centers by typing any non-period alphanumeric character
- 2) cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:
 - a) moving the cursor using the graphics controls, to a position where he wishes to bend the line,
 - b) typing a period (.),
 - c) repeating (a) and (b) until a maximum of five corners have been created.
 - d) completing the line segment from the last corner to symbol 2 by entering any non-period alphanumeric character.

If the user is on a VT100 terminal, a straight line connection is automatically created between the two nodes and stored in the database, but the screen remains unchanged.

If the user is in NODRAW mode on another terminal, the two options given above are still available. The only difference is that the connection lines are not displayed on the screen as the connection is defined.

6.3.16 ADE COMMAND: REDRAW

The REDRAW command causes the current architecture window to be redrawn to reflect any changes which have been made in NODRAW mode.

COMMAND SYNTAX:

REDRAW

RED

FUNCTION RESULT:

The display is redrawn to reflect the current architecture including all changes made by the user while in NODRAW mode.

6.3.17 ADE COMMAND: SAVE

The SAVE command copies the contents of the working database into the user's permanent database.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The permanent database is replaced with the contents of the working database, and the user is returned to the ADE ready state - # prompt. This command is useful when the user is defining a large system because it allows the user to protect the work done up to the point of issuing the SAVE command.

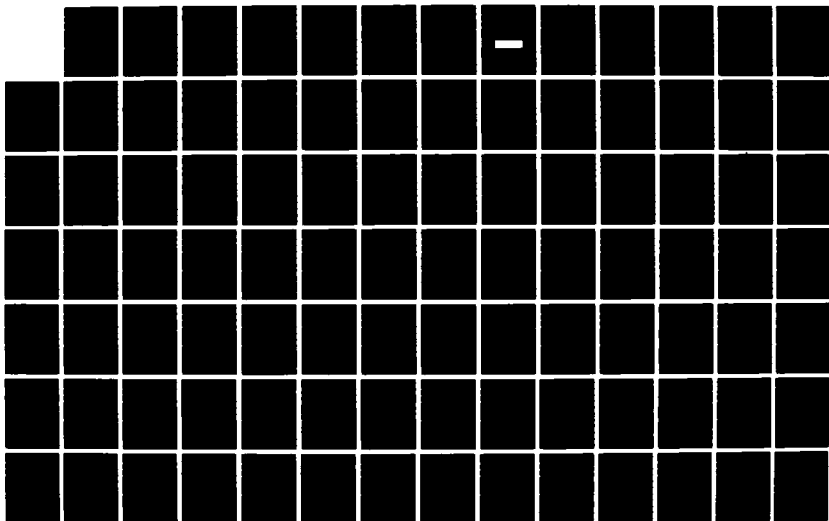
AD-A190 171

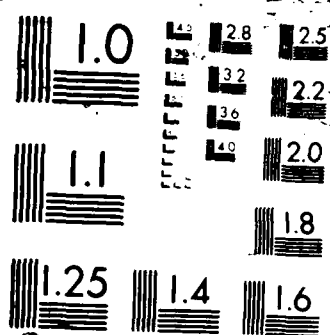
AUTOMATED INTERACTIVE SIMULATION MODEL (AISIM) VAX
VERSION 50 USER'S MANUAL (U) HUGHES AIRCRAFT CO
FULLERTON CA GROUND SYSTEMS GROUP V ALLERTON ET AL
29 MAY 87 ESD-TR-87-220 F19620-06-C-0070 F/G 12/3

3/4

UNCLASSIFIED

ML





6.3.19 Termination of an ADE Session

The ADE session is terminated by issuing the command,

END

This completes the edit portion of the ADE session and begins a sequence of events that leads to a return to the DUI Level. Before control is returned to the DUI level, however, the system gives the user the option of creating a new Legal Path Table. The Legal Path Table (LPT) created by the system is based upon the architecture that was created. The LPT consists of a two dimensional array. Entries in the array represent a means of getting from one node to another.

Entries contain two pieces of information:

- 1) the next node in the path from Node 1 to Node 2
- 2) the link used to get to the next node.

There are three basic methods of generating a Legal Path Table at the end of an ADE session. In response to the END command, the system questions the user:

BY WHICH METHOD DO YOU WISH TO GENERATE THE LPT (A, B, OR C)?

IF YOU HAVE AN ESTABLISHED LPT OR IF YOU WISH TO SKIP THIS STEP,
TYPE "END"

IF YOU DESIRE MORE INFORMATION ON METHODS A, B, OR C, TYPE "INFO"

After the pound sign (#) prompt, the user may enter either "A", "B", "C", "END", "HELP", or "INFO". If the user enters END and a carriage return after this or any subsequent # prompts without responding to the previous prompt question, any currently defined LPT, including none, will remain in effect, and control will return to the DUI level.

If any of the three options is chosen the, previously defined LPT will be deleted from the database and a new LPT will be produced. Since these algorithms may take several minutes, the user is provided with a message that lets him know the system is progressing with the LPT. The prompt initially reads "Generating LPT 1". After so many routes have been found, the message will change to "Generating LPT 2" and so on. The following paragraphs discuss the individual processing performed in response to methods A, B, and C.

METHOD A - Method A directly connects adjacent nodes in the architecture but no other paths are generated. This method is used when message routing paths are not of interest in the model. This method requires the least processing time to generate the LPT. After the user selects method A, the system will begin generation of the LPT. In general, AISIM will not solicit any further information if this method is used.

Method A detects two types of error. If the generator detects an unconnected node, the system will output the following error message:

UNREACHABLE NODE..."node name"

and control is transferred to the DUI level. If multiple links connect nodes, the system will prompt the user for resolution of ambiguous paths. The system will prompt with:

GOING FROM "Node name1" TO "Node name2" CAN GO

1. Through "next Node name" BY CHANNEL "channel name"
 2. Through "next Node name" BY CHANNEL "channel name"
- ENTER THE NUMBER OF THE ROUTE YOU WANT TO USE #

All "Through" options will be listed. The choice of path is selected by entering the number of the path after the pound sign (#) prompt. If there are ambiguous paths for other node pairs, the user will be prompted for resolution. If the user should ABORT the LPT generation the following prompt will be displayed:

UNABLE TO SAVE LPT

Control is then passed to the DUI level. If all ambiguities are clarified, the system will complete the generation of the LPT, and issue the following message:

SAVE OF LPT COMPLETE

The user is then at the DUI level.

METHOD B - Method B should be used when there is extensive routing through the architecture. Using Method B, AISIM will algorithmically find all possible legal paths through the system.

This can involve a lot of processing in fully connected architectures because a path from every node to every other node must be defined. For example, if there are 20 nodes then there will be 380 paths, 20 times 19.

The AISIM responses for method B are similar to those described in method A. Because AISIM will fully connect all nodes in the architecture there are bound to be many ambiguous paths. The user will be prompted to resolve all ambiguous paths.

METHOD C - Method C should be used when there is extensive routing through the architecture, also. Using Method C, AISIM will algorithmically find all possible legal paths through the system but will assume that the path for directly connected nodes in the architecture is the direct link. This can substantially decrease the number of paths the user must resolve.

The AISIM responses for method C are similar to those described in method A.

The HELP request causes the system to show the available commands.

The INFO request prints the following:

METHOD A defines as legal paths only connections directly between adjacent Nodes. Longer paths must be handled explicitly in the user Processes.

METHOD B generates all possible paths between each Node pair. You must identify default legal paths for each Node pair.

METHOD C generates all possible paths between each Node except for directly connected Nodes. In the case of adjacent Nodes, the direct connection is assumed as the legal path.

Type END and a carriage return to exit the LPT generation.

In figure 6-9, an AISIM architecture is shown. This architecture connects nine nodes together with 10 links. Using method A, the user is required to resolve 2 ambiguities. Using method B, the user is required to resolve 20 ambiguities. Using method C, the user is required to resolve 12 ambiguities. The Legal Path Tables using each of these methods is shown in figures 6-10 through 6-12.

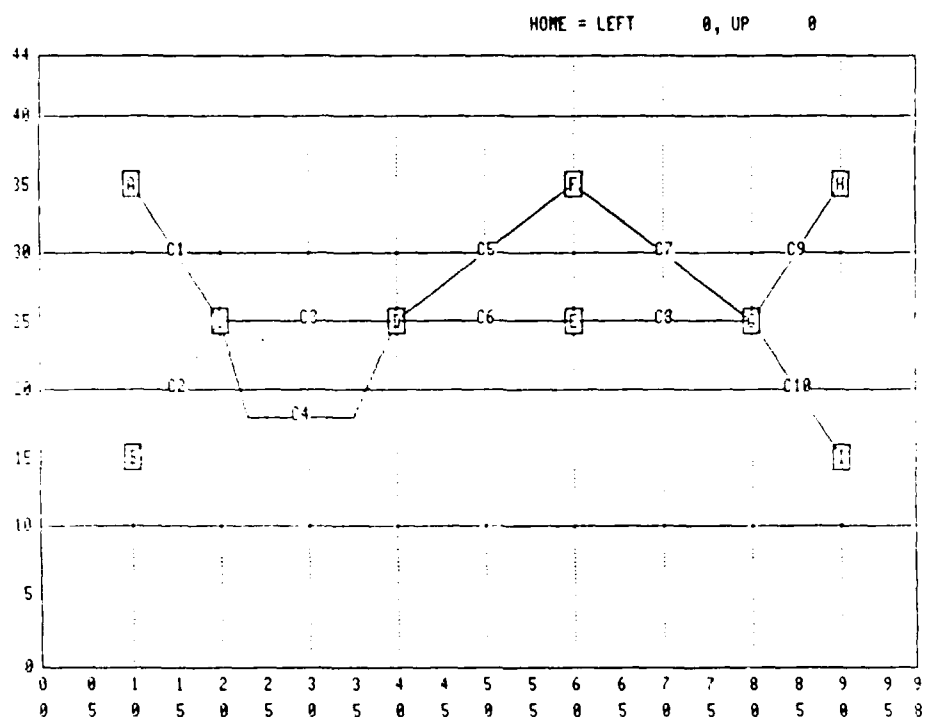


Figure 6-9. Sample Architecture

FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
D	C	C	C3
D	E	E	C4
D	F	F	C5
E	D	D	C6
E	F	F	C7
F	D	D	C5
F	E	E	C6
G	E	E	C8
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

Figure 6-10. Sample IPT Generated by Method A

LPT GENERATED FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	B	C	C1
A	C	C	C1
A	D	C	C1
A	E	C	C1
A	F	C	C1
A	G	C	C1
A	H	C	C1
A	I	C	C1
B	A	C	C2
B	C	C	C2
B	D	C	C2
B	E	C	C2
B	F	C	C2
B	G	C	C2
B	H	C	C2
B	I	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	D	C3
C	F	D	C3
C	G	D	C3
C	H	D	C3
C	I	D	C3
D	A	C	C4
D	B	C	C4
D	C	E	C8
D	E	E	C8
D	F	E	C8
D	G	E	C8
D	H	E	C8
D	I	E	C8
E	A	D	C8
E	B	D	C8
E	C	D	C8
E	D	F	C8
E	F	G	C8
E	G	H	C8
E	H	I	C8
F	A	D	C8
F	B	D	C8
F	C	D	C8
F	D	E	C8
F	E	G	C7
F	F	G	C7
F	G	G	C7
F	H	G	C7
F	I	G	C7
G	A	B	C8
G	B	B	C8
G	C	D	C8
G	D	E	C8
G	E	F	C7
G	F	H	C9
G	G	I	C10
G	H	I	C9
G	I	I	C9
G	A	C	C9
G	B	C	C9
G	C	C	C9
G	D	C	C9
G	E	C	C9
G	F	C	C9
G	G	C	C9
G	H	C	C9
G	I	C	C9
G	A	C	C10
G	B	C	C10
G	C	C	C10
G	D	C	C10
G	E	C	C10
G	F	C	C10
G	G	C	C10
G	H	C	C10
G	I	C	C10

Figure 6-11. Sample LPT Generated by Method B

LPT GENERATED FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	B	C	C1
A	C	C	C1
A	D	C	C1
A	E	C	C1
A	F	C	C1
A	G	C	C1
A	H	C	C1
A	I	C	C1
B	A	C	C2
B	C	C	C2
B	D	C	C2
B	E	C	C2
B	F	C	C2
B	G	C	C2
B	H	C	C2
B	I	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	D	C3
C	F	D	C3
C	G	D	C3
C	H	D	C3
C	I	D	C3
D	A	C	C4
D	B	C	C4
D	C	C	C4
D	E	E	C6
D	F	E	C6
D	G	E	C6
D	H	E	C6
D	I	E	C6
E	A	D	C5
E	B	D	C5
E	C	D	C5
E	D	D	C5
E	E	D	C5
E	F	D	C5
E	G	D	C5
E	H	D	C5
E	I	D	C5
F	A	D	C6
F	B	D	C6
F	C	D	C6
F	D	D	C6
F	E	D	C6
F	F	D	C6
F	G	D	C6
F	H	D	C6
F	I	D	C6
G	A	D	C7
G	B	D	C7
G	C	D	C7
G	D	D	C7
G	E	D	C7
G	F	D	C7
G	G	D	C7
G	H	D	C7
G	I	D	C7
H	A	D	C8
H	B	D	C8
H	C	D	C8
H	D	D	C8
H	E	D	C8
H	F	D	C8
H	G	D	C8
H	H	D	C8
H	I	D	C8
I	A	D	C9
I	B	D	C9
I	C	D	C9
I	D	D	C9
I	E	D	C9
I	F	D	C9
I	G	D	C9
I	H	D	C9
I	I	D	C9

Figure 6-12. Sample LPT Generated by Method C

SECTION 7

ANALYSIS USER INTERFACE (AUI)

After completing a model design using the DUI, the model can be exercised using the commands available in the AUI. During simulation, statistics are kept on Variable values, Item throughput, Resource utilization, queueing delays, Queue lengths, Action times, Process execution, and Process timing. A set of output reports organizes these statistics for printing off line or viewing on-line (while in the AISIM READY level) after completion of the simulation run. Plots of selected model parameters, however, may be drawn on the screen when simulation is halted at a breakpoint, end of period, or end of simulation.

When the user initiates an Analyze session, the user is given an opportunity to enter a description of the current simulation run. The description will be placed at the top of the output report and at the beginning of the statistics portion of the output report. This descriptive text is stored in the user's design data base. If the user wishes to describe the current run, a form is displayed for the user to enter the text. If a previous description exists, it is displayed in the form. The form for entering the description is shown in figure 7-1.

ENTER DESCRIPTION OF MODEL:

The image shows a dark, rectangular area that appears to be a scan of a form. It is mostly black with some faint, illegible white markings, suggesting it was a form for entering text but the content is not legible in this scan.

Figure 7-1. Form for Text Description

The current date and time the simulation is run are also placed at the top of the output report and at the beginning of the statistics portion of the report.

The command issued to enter the AUI from the AISIM READY level contains an optional parameter NOXLATE. If this parameter is omitted, the project database is first translated before a simulation is performed. This translation converts the database into the format required for simulation execution.

If the NOXLATE parameter is used, no translation will take place. The last translation of the project database is used in executing a simulation run. Since another translation is required only if the database was changed (in the DUI) since the last translation, it is not always necessary to repeat the translation process at the start of an analysis session. The NOXLATE option permits skipping of the translation step.

In the translating process, the user is asked the following question, if there is more than one Scenario in the project database:

WHICH SCENARIO DO YOU WISH TO TRANSLATE?

The user must respond with a valid Scenario name, one that has been defined previously in DUI level. A carriage return in response to this question will cause AISIM to list available Scenarios.

If the Scenario name given is invalid the system will respond:

INVALID SCENARIO NAME - REENTER

The user should then enter the correct Scenario name.

When translation of the model and Scenario has completed, the simulator reads the translated database and checks for errors. If the simulator detects one or more errors, the message

ERRORS DETECTED IN MODEL TRANSLATION

is displayed, the AUI level is exited and the user is returned to the AISIM READY level.

At this point the user should enter the EDIT command (described in section 13.3). This automatically invokes the EDT line editor on the project report file. The user should use the Find command of the VAX/VMS EDT editor to list all occurrences of "####". This will result in a list of all errors detected during initialization. Each error documents a problem detected in the model. The EDT line commands used to view the report file are discussed in section 13.3.

If no errors are detected, the following message is displayed:

NO ERRORS DETECTED IN MODEL TRANSLATION
YOU MAY NOW ENTER COMMANDS

The system provides a # prompt and is ready to accept any of the valid AUI commands. These commands are described in the following pages.

During each of the three phases of analysis - 1) pre-simulation (before the first GO command is issued), 2) mid-simulation (after the first GO command is issued but before simulation termination), and 3) post-simulation (after simulation termination), the user can invoke different commands.

PRE-SIMULATION COMMANDS:

CANBREAK DEFLOT EDIT END GET GO

HELP INFREG LIST LISTVAL UNITS

SAVE (plot definitions) SETBREAK DELETE

MID-SIMULATION COMMANDS:

CANBREAK EDIT END G^o HELP LIST LISTVAL
PLOT SAVE SETBREAK UNITS

POST-SIMULATION COMMANDS:

END HELP LIST LISTVAL PLOT SAVE UNITS

The simulation is started with the GO command.

The SETBREAK and CANBREAK commands are used to establish and cancel stopping conditions (or breakpoints) for the simulation. EDIT is used to make temporary changes to Constants, Variables, and the random number seed values (the keyword is STREAM) upon which stochastic timing and probabilistic branching are based. The Scenario and Loads may be modified by changing the values of parameters specified by Constants. A limited number of Resources in the model sometimes causes a bottleneck which is evidenced by a waiting line or queue. The effects of this queueing may be eliminated by changing the available Resources to an unlimited quantity. The INFRES command is used to do this on a temporary basis. LIST and LISTVAL are used to display model entities, their attributes, and their values. LISTVAL also allows the user to examine the current random number seeds. The END command returns control to the AISIM READY level.

The DEF PLOT and PLOT commands are used to specify what information is to be gathered for graphs and to request the graph to be displayed at the terminal, respectively. The DEF PLOT command can only be used prior to the start of simulation since the simulator must know what statistics to sample. The UNITS command is used to specify the time units for output produced by the simulation.

A simulation may be performed in periods and is suspended at the end of the number of periods specified. As each period completes, a message is displayed on the screen indicating the period number just completed and the simulation and real clock time at which it completed. The number of periods to be simulated is specified as an optional parameter of the GO command. The user is prompted at the end of the period with the message:

END OF PERIOD
YOU MAY NOW ENTER COMMANDS

and with an audible 'beep' at the terminal.

The user can now make changes in the values of Variables, set breakpoints, display plots, or cancel breakpoints. By suspending the simulation at the end of a period, the user can dynamically interface with the model.

A similar result occurs at a user specified breakpoint, except that the message reads:

BREAK POINT REACHED:
(description of the condition of the breakpoint)
YOU MAY NOW ENTER COMMANDS

An audible 'beep' is also sounded at this point.

The AUJ level commands are described in detail on the following pages.

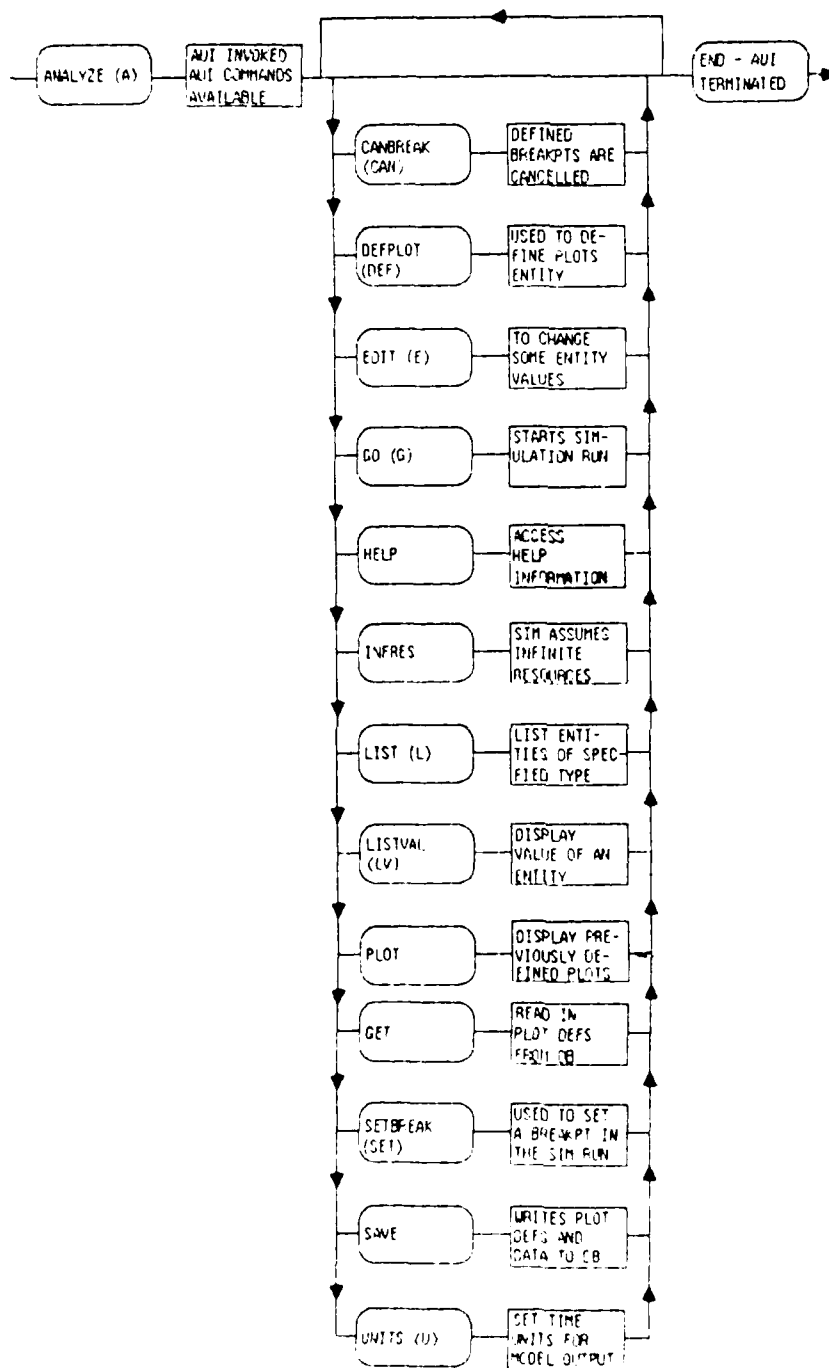


Figure 7-2. Analysis User Interface Commands

CANBREAK
CAN

DEFPLOT {entity-type},{entity-name},...,{entity-name}
DEF

DELETE TITLE,{titlenum1},...,{titlenumn}
DELETE TITLE,*
DEL

EDIT {entity-type},{entity-name},{new-value}
E

END

GET DEF,{setname}

GO [n]
G

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

INFRES {entity-name},...,{entity-name}
*

LIST {entity-type}
LIST DEF
LIST PLOT
LIST TITLE
L

LISTVAL {entity-type},{entity-name}
LV

PLOT

SAVE {settype},{setname},{descr}

SETBREAK {entity-type},{entity-name},{rel-oper},{value}
SET

UNITS {unit-type}
U

Figure 7-3. AUI Command Summary

7.1 AUI COMMAND: CANBREAK

The CANBREAK command allows the user to cancel a previously defined breakpoint. See the SETBREAK command in section 7.14.

COMMAND SYNTAX:

CANBREAK

CAN

FUNCTION RESULTS:

A previously defined breakpoint is canceled.

7.2 AUI COMMAND: DEFLOT

DEFLOT is a pre-simulation command that allows the user to specify what plot data to collect over the period of simulation. The specified plot is added to the present set of plot specifications. This plot data is later graphed with the use of the PLOT command.

COMMAND SYNTAX:

```
DEFLOT {entity-type},{entity-name},...,{entity-name}
```

```
DEF
```

where:

{entity-type} is a required parameter indicating a valid entity-type (i.e., Variable, Queue, Resource, Process, Item).

{entity-name} is a required parameter indicating the name of the entity whose value is to be plotted. The user can enter a list of entity names (up to a maximum of eighty characters) of the given entity type at a time. Multiple DEFLOT commands can be used to define more plots.

FUNCTION RESULT:

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic. The list of statistics displayed depends on the entity-type and attribute selected.

If only one choice for either an attribute or a statistic exists, the form is not displayed. The forms displayed are shown in figures 7-4 through 7-9. A sample plot is shown in figure 7-9. After the simulation has generated plot data, the plots can be displayed at the user's graphics terminal using the PLOT command (see section 7.12) and printed on a graphics printer (see appendix A.4).

A maximum of ten plots may be specified during any Analysis session.

ATTRIBUTES (PLACE AN X NEXT TO ONLY ONE)

NUMBER CREATED
NUMBER DESTROYED
NUMBER IN SYSTEM
TIME IN SYSTEM

Figure 7-4. DEFPLOT Form for Items

ATTRIBUTES (PLACE AN X NEXT TO ONLY ONE)

COMPLETION TIME
AUTO SCHEDULED
CALL SCHEDULED

STATISTICS (PLACE AN X NEXT TO ONLY ONE)

CURRENT
CUMULATIVE MEAN
CUM STANDARD DEV
CUMULATIVE MIN
CUMULATIVE MAX
PERIOD MEAN
PER STANDARD DEV
PERIOD MIN
PERIOD MAX

Figure 7-5. DEFPLOT Forms for Process

ATTRIBUTES (PLACE AN X NEXT TO ONLY ONE)

NUMBER IN QUEUE
NUMBER BLOCKED
TIME IN QUEUE
TIME BLOCKED

STATISTICS (PLACE AN X NEXT TO ONLY ONE)

CURRENT
CUMULATIVE MEAN
CUM STANDARD DEV
CUMULATIVE MIN
CUMULATIVE MAX
PERIOD MEAN
PER STANDARD DEV
PERIOD MIN
PERIOD MAX

Figure 7-6. DEFPLOT Forms for Queues

ATTRIBUTES (PLACE AN X NEXT TO ONLY ONE)

IN WAIT QUEUE
IN BUSY QUEUE
IN IDLE QUEUE
WAIT TIME
BUSY TIME
REQUEST TIME

STATISTICS (PLACE AN X NEXT TO ONLY ONE)

CURRENT
CUMULATIVE MEAN
CUM STANDARD DEV
CUMULATIVE MIN
CUMULATIVE MAX
PERIOD MEAN
PER STANDARD DEV
PERIOD MIN
PERIOD MAX

Figure 7-7. DEFPLOT Forms for Resources

STATISTICS (PLACE AN X NEXT TO ONLY ONE)

CURRENT
 CUMULATIVE MEAN
 CUM STANDARD DEV
 CUMULATIVE MIN
 CUMULATIVE MAX
 PERIOD MEAN
 PER STANDARD DEV
 PERIOD MIN
 PERIOD MAX

Figure 7-8. DEFPLOT Form for Variables

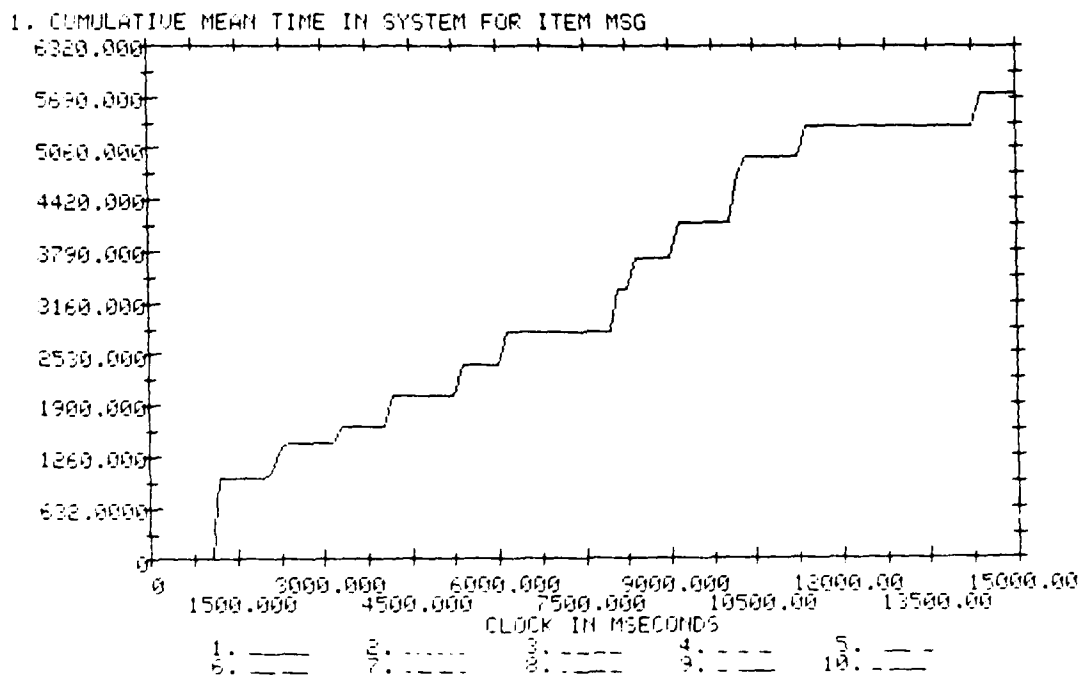


Figure 7-9. Sample Plot

7.3 AUI COMMAND: DELETE

DELETE is a pre-simulation command that allows the user to delete plot definitions which were set up through the DEF PLOT or GET commands.

COMMAND SYNTAX:

```
DELETE TITLE,{titlenuml},...,{titlenumn]
```

```
DELETE TITLE,*
```

```
DEL
```

where:

{titlenuml} is a required parameter indicating the number of the plot definition to be deleted. A list of definition numbers may be entered.

* is a literal parameter, indicating all of the current plot titles.

FUNCTION RESULT:

This command causes the specified plot definitions to be deleted from those being used for the current simulation run. If the definitions came from a definition set in the database, they still reside in that set in the database, i.e., the database is not modified by this command.

This command allows a user to retrieve plot definitions with the GET DEF command (see section 7.6) and then use only selected definitions for a particular simulation run.

The user can see the numbers of the current plot definitions with the LIST TITLE command (see section 7.10).

7.4 AUI COMMAND: EDIT

The EDIT command in the Analysis mode allows the user to change the value of either a Constant, a Variable, or specification of the random number stream used to represent probabilistic events. The value of a Variable with an alpha literal as its initial value cannot be changed with this command.

COMMAND SYNTAX:

```
EDIT {entity-type},{entity-name},{new-value}
```

```
E
```

where:

{entity-type} is a required parameter indicating which type of entity is to be changed (either Constant, Variable, or Stream).

{entity-name} is a required parameter indicating the name of the Constant, Variable or Stream (Branch, Load, or Action) which is to be changed.

{new-value} is a required parameter indicating the new value of a Constant or Variable or for STREAM, the new random number stream. The new value may be expressed in one to twelve digits, and includes the value zero. The legal values of "new value" when specifying a random number stream are 1 through 10.

NOTE: Constants may be changed only before the start of the first simulation period. Variables and Streams may be changed before the start of any simulation period or at a breakpoint.

FUNCTION RESULT:

The value of the Constant or Variable or Stream is changed to the new value, and remains at that value until changed by another EDIT command. This command only affects the current translation of the database; therefore, at the end of an Analysis session the Constant or Variable or Stream is restored to its original value.

If the value of the Stream is not changed, default values are:

Action: 3, for random Action durations

Branch: 2, for the PROB Primitive

Load: 1, for random intervals between a Load's triggering of another Process instance.

AUI / END

7.5 AUI COMMAND: END

The END command is used to terminate an Analysis session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

This command causes all displays to be cleared and, if plots were generated, asks the user "Do you wish to save plot definitions? (Y/N)" and "Do you wish to save plot data? (Y/N)". If the answer to a question is yes, the user is prompted for the required information before control returns to the AISIM READY level. Plot data and definitions are stored in a file called project.PLT where project is the name of the user's project database. Upon termination of an Analysis session, a copy of the output report is automatically printed.

7.6 AUI COMMAND: GET

The GET command allows the user to retrieve a previously saved set of plot definitions and add them to the current plot specification. The plot specification defines what plot data will be collected during the simulation. The LIST DEF command may be used to obtain a list of the available plot definition sets.

COMMAND SYNTAX:

```
GET DEF,{setname}
```

where:

{setname} is the name of the set containing the plot definitions. The GET command may be issued only before the first simulation period.

FUNCTION RESULT:

The set of plot definitions is retrieved and made a part of the current set to be used by the Analysis function.

The LIST DEF command (see section 7.10) may be used to obtain a list of the available plot definition sets.

7.7 AUI COMMAND: GO

The GO command allows the user to start or resume a simulation run.

COMMAND SYNTAX:

GO [n]

G

where:

[n] is an optional parameter that specifies how many periods the simulation is to run. If not given, the default result is that the entire simulation defined by the selected Scenario is executed. If an n greater than the number of periods specified in the Scenario is entered, the simulation executes all periods specified in the Scenario and no more.

FUNCTION RESULT:

This command, which is valid before any simulation period or at a breakpoint, begins or resumes the simulation of the translated Scenario.

If used to resume the simulation, resumption occurs at the breakpoint or at the beginning of the next simulation period.

7.8 AUI COMMAND: HELP

The HELP command provides the user with access to help information about the current AUI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the AUI function is displayed. The commands acceptable at the current AUI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the AUI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

7.9 AUI COMMAND: INFRES

The INFRES command causes the simulation to assume the existence of infinite available Resources for specified Resources.

COMMAND SYNTAX:

INFRES {entity-name},...,[entity-name]

INFRES *

where:

{entity-name} is a required parameter indicating the name of the Resource for which unlimited units are available. A list of up to eight Resource names at a time may be entered.

* is a literal parameter, indicating all Resource entities in the model are to be assumed to have unlimited units.

The INFRES command can be entered multiple times to set infinite resources for more entities.

FUNCTION RESULT:

This command, which is only valid before the start of the first simulation period, allows the assumption that infinite Resources are available for the specified Resources during the Scenario being simulated.

7.10 AUI COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name and its description.

COMMAND SYNTAX:

LIST {entity-type}

LIST PLOT

LIST DEF

LIST TITLE

L

where:

{entity-type} is a required parameter indicating any of the specific model entities listed below.

ENTITY	ABBREVIATION
CONSTANT	C
RESOURCE	R
PROCESS	P
VARIABLE	V
ITEM	I
QUEUE	Q

This command is valid at any time during an Analysis session.

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type. If the argument is PLOT, a list of saved plot sets is given. If the argument is DEF, a list of the saved plot definition sets is given. If the argument is TITLE, a list of the plots defined for the current simulation is given.

7.11 AUI COMMAND: LISTVAL

The LISTVAL command allows the user to display the current statistics for the named entity.

COMMAND SYNTAX:

```
LISTVAL {entity-type},{entity-name}
```

```
LV
```

where:

{entity-type} is a required parameter indicating a valid Analysis system entity-type. The valid entity types are the following:

Clock	
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Stream	S
Variable	V

{entity-name} is a required parameter indicating the name of the entity whose value is to be listed. When requesting the Stream for Loads, Branches or Actions, or the Clock, this field is omitted.

FUNCTION RESULT:

The name of the entity requested is printed out with a listing of all statistics for that entity.

The prompt "**** Enter YES/Y to continue, NO/N to abort ****" is displayed. If the user wishes to end the LISTVAL listing, "NO" is entered and the AUI READY prompt is displayed. If "YES" is entered the next page of the listing is displayed, if there is one, with the prompt displayed again. If there is no further data to be displayed, the user is returned to the AUI ready level.

7.12 AUI COMMAND: PLOT

The PLOT command allows the user to produce a graph of the plot data collected during the simulation.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

This command, which is valid at the end of a simulation period or at a breakpoint, causes the display of a form containing the plot titles which were defined using the DEFLOT command (see section 7.2). From this form the user may select any, all, or none of the listed titles.

When the selected titles have been entered, the user is presented with the plot grid. The selected plots are produced and the user is prompted for more Analysis mode commands.

Each of the plots is produced in a unique line pattern.

Once displayed on the terminal, graphs can be transferred to a hardcopy device (see appendix A.4). An example of the form that is displayed to allow the user to select a plot is shown in figure 7-10. A sample plot is shown in figure 7-11.

NOTE: Due to limits imposed by graphics screen resolution, only a sample of the data points produced by the simulation are included in the plot (see appendix A.3).

PLACE AN X NEXT TO THE TITLES YOU WISH TO PLOT

CUMULATIVE MEAN # IN WAIT QUEUE FOR RESOURCE B6B1
 CURRENT VALUE FOR VARIABLE U_ROUTER
 CUMULATIVE MEAN COMPLETION TIME FOR PROCESS MRS
 CUMULATIVE MEAN TIME IN SYSTEM FOR ITEM MSG
 CUMULATIVE MEAN WAIT TIME FOR RESOURCE B1B2
 CUMULATIVE MEAN WAIT TIME FOR RESOURCE B2B3

Figure 7-10. Sample Form for Selecting Plots

1. CUMULATIVE MEAN WAIT TIME FOR RESOURCE B1B2
2. CUMULATIVE MEAN WAIT TIME FOR RESOURCE B2B3

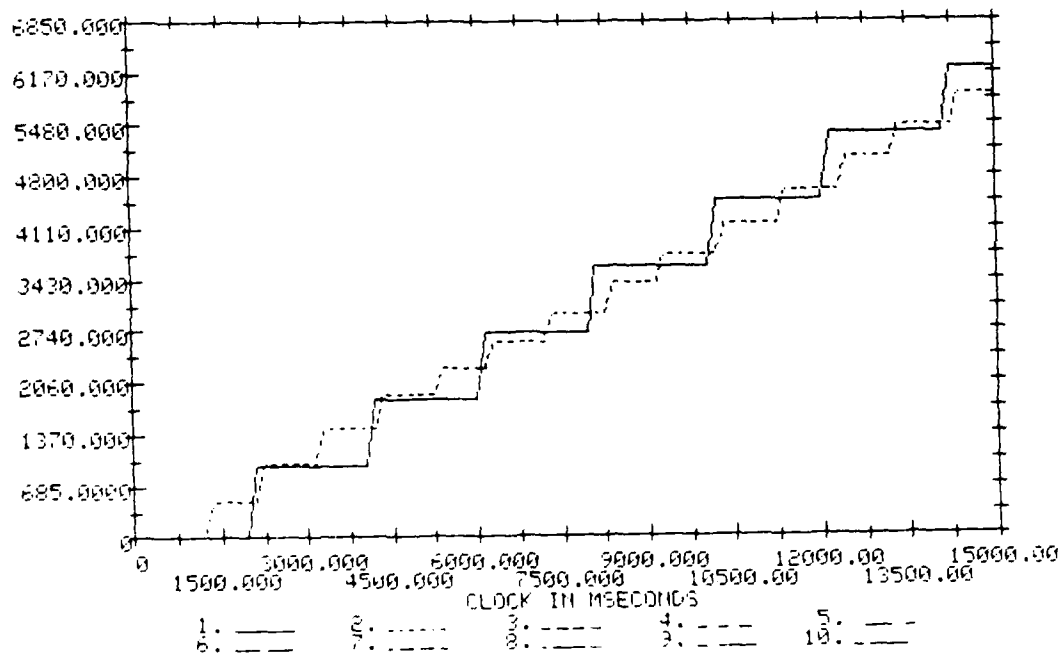


Figure 7-11. Sample Plot

7.13 AUI COMMAND: SAVE

SAVE is used to save current plot definitions or plot data and transfer them to the Analysis database.

COMMAND SYNTAX:

SAVE {settype},{setname},{description}

where:

{settype} is

1. DEF to save plot definitions, or
2. PLOT to save plot data.

{setname} 1 to 8 character name to be given to the set.

{description} is a description of the set.

FUNCTION RESULT:

Plot definitions or plot data are flagged to be saved in the Analysis database when the Analysis session is terminated. If {setname} already exists, the user is queried to reuse the old set. A "yes" response will replace the old set with the new set. A "no" response will cause a prompt for a new set name.

7.14 AUI COMMAND: SETBREAK

The SETBREAK command allows the user to set a single breakpoint in the simulation run that is executed when a defined relationship has been satisfied.

COMMAND SYNTAX:

```
SETBREAK {entity-type},{entity-name},{rel-oper},{value}
```

```
SET
```

where:

{entity-type} is a required parameter indicating which type of entity is to be tested (Variable, Resource, or Process).

{entity-name} is a required parameter indicating the name of the entity to be tested.

{rel-oper} is a required parameter indicating the relational operator (EQ, NE, LE, GT, GE, LT) of the test.

{value} is a required parameter used to set the value for which the named entity is to be tested. This value may be expressed in one to twelve digits, and includes the value zero.

FUNCTION RESULT:

A breakpoint is usually used in verification of a model or to examine Variable values. Typically, a simulation run executes start to finish and does not allow the user to examine the simulation state at specific times during simulation. The breakpoint allows the user to halt the simulation and examine its state based upon the value of some system element.

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic.

If there is only one choice for either an attribute or a statistic, the form is not displayed. Attribute and statistic forms are shown in figures 7-5, 7-7, and 7-8.

This command is valid at the beginning of a simulation period or at a breakpoint.

When a breakpoint is reached, it is automatically cleared.

7.15 AUI COMMAND: UNITS

The UNITS command is used to change the time units that are used in the display of output data from the simulation, namely plot diagrams, the output report and trace output.

COMMAND SYNTAX:

UNITS {unit-type}

U

where:

{unit-type} is a required parameter indicating the time units to be used. The valid entries for this command and their abbreviations and meanings are as follows:

<u>Command entry</u>		<u>Meaning</u>
nseconds	(ns) -	nanoseconds
useconds	(us) -	microseconds
mseconds	(ms) -	milliseconds
seconds	(s) -	seconds
minutes	(m) -	minutes
hours	(h) -	hours
days	(d) -	days

FUNCTION RESULT:

The output and trace report will be written using the time units most recently specified by the user before the report was generated. Normally these are the units specified at the start of the simulation. If the user does not make use of the UNITS command, the units specified as OUTPUT UNITS in the Scenario entity will be used. The user can change the units used to display plot data anytime during the AUI session; i.e., the UNITS command will always affect the plot display, but will only affect the written reports if they have not been generated yet.

7.16 TERMINATION OF AN AUI SESSION

An AUI session is terminated and control is transferred to the AISIM READY level through the command:

END

FUNCTION RESULT:

When the END command is issued, any plot data or plot definitions which the user saved during the AUI session are placed in the user's Analysis database. Any attempts to reuse plot data or plot definition sets are resolved at this time. The user is then returned to the AISIM READY level.

SECTION 8

REPLOT USER INTERFACE (RUI)

The Replot User Interface (RUI) allows the user to:

- (1) plot data saved from previous analysis runs,
- (2) to delete old plot data and plot definition sets from the data base.
- (3) create new plot data sets from data previously saved in separate plot data sets.

When plot data is retrieved from the Analysis database via the GET command (see section 8.4), the plot data is stored in a temporary plotset. This temporary plotset exists for the current Replot session only. Data from different Analysis runs may be retrieved from the database. All of the data is then stored together in the temporary plotset, and may be plotted on the same graph. The SAVE command (see section 8.8) will store all of the data in the temporary plotset into a new, permanent plotset in the analysis database. The temporary plotset can be cleared out (i.e., plot data in it is deleted) using the CLEAR command (see section 8.1). The CLEAR command does not affect permanent data stored in the database.

Once displayed on the terminal, plots can be transferred to a hardcopy device (see appendix A.4).

The RUI level commands are described in detail on the following pages.

CLEAR

DELETE {settype},{setname}
DEL

END

GET PLOT,{set-name}

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST {entity-type}
L

PLOT

SAVE PLOT,{set-name},{description}
S

UNITS {unit-type}
U

Figure 8-1. RUI Command Summary

RUI / CLEAR

8.1 RUI COMMAND: CLEAR

CLEAR is used to delete all of the plot data from the temporary plotset and to clear the screen.

COMMAND SYNTAX:

CLEAR

FUNCTION RESULT:

The temporary plot data set is emptied, and the screen is cleared. Plots saved in the database are unaffected.

8.2 RUI COMMAND: DELETE

DELETE is used to delete a set of plot definitions or plot data from the Analysis data base.

COMMAND SYNTAX:

```
DELETE {settype},{setname}
```

```
DEL
```

where:

{settype} is:

DEF to delete plot definitions, or

PLOT to delete plot data.

{setname} is the name of the set to be deleted.

FUNCTION RESULTS:

The specified set of plot data or plot definitions are deleted from the Analysis data base. The current temporary plot set is unaffected.

RUI / END

8.3 RUI COMMAND: END

END is used to exit the RUI.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The user is returned to the AISIM READY level.

3.4 RUI COMMAND: GET

GET is used to retrieve plots from a set of plot data in the data base and to make it part of the current set of plots to be displayed by the PLOT command.

COMMAND SYNTAX:

```
GET PLOT,{set-name}
```

where:

{set-name} is the name of the set containing the desired plot data.

FUNCTION RESULT:

The set of available plots is displayed. The user is then prompted for the plot(s) to be retrieved for use by the PLOT command.

The names of the plot data sets may be listed using the LIST command (see section 8.6).

8.5 RUI COMMAND: HELP

The HELP command provides the user with access to help information about the current RUI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the RUI function is displayed. The commands acceptable at the current RUI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the RUI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

8.6 RUI COMMAND: LIST

LIST is used to list all entities of the specified type.

COMMAND SYNTAX:

LIST {entity-type}

where:

{entity-type} is a required parameter indicating a valid entity type. It can be one of the following:

DEF to list plot definition sets

PLOT to list plot data sets

TITLE to list current plot titles

FUNCTION RESULTS:

Names of all entities of the requested type are displayed.

8.7 RUI COMMAND: PLOT

The PLOT command is used to display a plot of the activity of an entity.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

The set of available plots is displayed. The user is then prompted for the plot(s) to be graphed.

When the selected plot titles have been entered, the appropriate plot is displayed. Each of the plots is produced in a unique line pattern. If only one plot is defined, it will be displayed with no prompting.

Once displayed on the terminal, plots can be transferred to a hardcopy device (see appendix A.4).

8.8 RUI COMMAND: SAVE

The SAVE command is used to save the data that is being held in the current temporary plot data set into a permanent plot data set in the database.

COMMAND SYNTAX:

```
SAVE PLOT,{set-name},{description}
```

```
S
```

where:

{set-name} is a 1 to 8 character name to be given to the set

[description] is a description of the set

FUNCTION RESULT:

The plot data contained in the temporary plot data set (as a result of previous GET PLOT commands) is saved into the new plot data set. This command enables the user to combine plots from various simulation runs into a single plot data set.

8.9 RUI COMMAND: UNITS

The UNITS command is used to change the time units used in displaying plot data.

COMMAND SYNTAX:

UNITS {unit-type}

U

where:

{unit-type} is a required parameter indicating the time units to be used. The valid entries for this command and their abbreviations and meanings are as follows:

<u>Command entry</u>	<u>Meaning</u>
nseconds (ns) -	nanoseconds
useconds (us) -	microseconds
mseconds (ms) -	milliseconds
seconds (s) -	seconds
minutes (m) -	minutes
hours (h) -	hours
days (d) -	days

FUNCTION RESULT:

The time units used during the display of plot data are changed to the specified value.

SECTION 9

HARDCOPY USER INTERFACE (HUI)

The Hardcopy User Interface (HUI) is used to plot the flowcharts for one, several, or all Processes in the specified project database. In order for the Hardcopy Function to be exercised, the following conditions must be in effect:

For an HP2647 terminal:

1. An HP2631G Graphics Printer must be connected to the HP2647A Graphics Terminal with the HP-IB communications bus.
2. The HP-IB bus address of the printer must be set to one (1).
3. The printer must be turned on and set to ON LINE mode.
4. For proper formatting, the length of the paper in the printer must be either 8 1/2 inches or 11 inches long.

For a TEK4105 terminal:

1. A TEK4695 graphics copier must be connected to the TEK4105 terminal.

For an HP2623 terminal:

1. The internal printer must be functional.

The HUI is entered from the AISIM READY level by typing the command:

```
HCOPI [PROJECT(project)] [TERM(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database with the Processes of interest. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified. The valid terminal types are the following:

HP - HP2647A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal

The first information that the HUI requests is:

PLOT ALL THE PROCESSES IN DATABASE? (YES OR NO)

The user responds with "NO" to specify selected Processes for plotting. A "YES" response will cause the system to automatically plot all of the Processes contained in the project data base.

The system then requests information about the printing medium for an HP2647A terminal:

ENTER PRINTER PAGE SIZE (A/B):

A) 8 1/2 INCHES

B) 11 INCHES.

LENGTH=

Depending on the paper in the graphics printer, the user responds by entering "A" or "B". This information is used by the HUI to center the Process graphics on the page and to insure correct form feeding. Entering any other option besides "A" or "B" causes the prompt to be reissued.

The user is then instructed to:

POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.

By doing this, the user sets up the proper alignment of the paper in the printer and initiates execution of the Hardcopy plotting software.

When the carriage return has been entered, the HUI begins the plotting procedure by initializing the HP2631G printer with the correct form information. This initialization is usually characterized by a rapid movement of the print head.

If the user is on an HP2623 terminal, only the following prompt occurs to start the Hardcopy operation:

POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.

When the user presses the carriage return, AISIM initiates execution of the Hardcopy plotting software.

Note: the following terminal configuration must be set up on the HP2623 terminal in order to run the Hardcopy program. These settings need to be set only once unless their configuration is changed at some later time. First press the following function keys:

<AIDES>

<CONFIG KEY>

<DATACOMM CONFIG>

Then tab through the form display to the

RECV PAGE

field. Press the NEXT key until the field reads

XON XOFF

Then tab to the

XMIT PAGE

field. Press the following two keys:

<CONFIG KEY>

<TERMINAL CONFIG>

Tab through the form to the

INHNDSHK(G)

key and press the NEXT key until the field reads

YES

The terminal is now set up for the Hardcopy function.

If the user is on a TEK4105 terminal, the following information is requested:

ENTER PRINTER PAGE SIZE (A/B):

A) 8 1/2 INCHES

B) SMALLER COPY SIZE

LENGTH=

This information is used to create standard size flow diagrams or reduced size diagrams.

The user is then instructed to:

POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.

When the user presses the carriage return, AISIM initiates execution of the Hardcopy plotting software.

If the user has requested automatic plotting of all of the Processes, they are plotted in alphabetic order.

If the user asked to select Processes, the following prompt is given:

PROCESS NAMES TO PLOT: (CR TO EXIT)

The user then supplies the name of the Process he wishes plotted, followed by a carriage return. The Process is plotted and the HUI responds with:

<Process name> PLOTTED

The system will then give the selection prompt again for another Process to be plotted. The user continues entering Process names one at a time, followed by a carriage return, or exits the HUI by entering a carriage return only.

The way in which the HUI plots a Process in either of the two modes described above is as follows:

1. The first screen of Primitives in the Process are painted on the screen of the terminal.
2. The Process name is written at the top of the page.
3. If the first page of the Process is being plotted, the Process description is also written across the top of the page.
4. The Process graphics are transferred from the terminal screen to the page in the printer and a form feed is generated.
5. If there are no more Primitives in the Process, the plotting is terminated for the Process; otherwise, the terminal screen is erased, and next six primitives are painted on the screen, and steps 2 through 5 are repeated.

When the HUI has finished plotting all of the requested Processes, the message "ALL DONE" is printed and the user is returned to the AISIM READY level.

SECTION 10

LIBRARY USER INTERFACE (LUI)

The Library User Interface allows the user to do the following:

1. Move entities from a project database into a storage area called a "buffer" using the MERGEOUT sublevel of the LUI.
2. Move entities from a buffer into the database of another project using the MERGEIN sublevel of the LUI.
3. Move entities from a buffer into a library of entities using the CHECKIN sublevel of the LUI.
4. Move entities from a library to a buffer using the CHECKOUT sublevel of the LUI.
5. Convert a version 3.0 or 4.0 project database to a version 5.0 compatible project database.

Two libraries are available. One is a user library in which a user can place entities for private use. Another is an AISIM system library which contains models available for public use. Models are groups of AISIM entities which represent some function or group of functions (see the message routing submodel, appendix D). There are restrictions on the placement of entities in the system library because it is desirable to insure that the public models are not lost or tampered with. For this reason, general users cannot modify the AISIM system library. Access is restricted to the AISIM administrator.

The LUI sublevel is accessible from the AISIM READY level by issuing the command:

LIBRARY

The system will then respond with the prompt:

LIBRARY READY

and the user may invoke any of the five LUI sublevels listed in the LUI Command Summary figure 10-1. Figure 10-2 shows the actions of the various LUI functions.

CHECKIN	[BUFFER(buffer)]	[LIBRARY(library)]	[TERM(terminal)]
CI	[B(buffer)]	[L(library)]	[T(terminal)]
CHECKOUT	[BUFFER(buffer)]	[LIBRARY(library)]	[TERM(terminal)]
CO	[B(buffer)]	[L(library)]	[T(terminal)]
CONVERT	[PROJECT(project)]	[TERM(terminal)]	DBVERS(version)
CONV	[P(project)]	[T(terminal)]	DB(version)
HELP	[subtopic],..., [subtopic]		
HELP	[@topic], [topic-name], [subtopic],..., [subtopic]		
MERGEIN	[PROJECT(project)]	[BUFFER(buffer)]	[TERM(terminal)]
MI	[P(project)]	[B(buffer)]	[T(terminal)]
MERGEOUT	[PROJECT(project)]	[BUFFER(buffer)]	[TERM(terminal)]
MO	[P(project)]	[B(buffer)]	[T(terminal)]

Figure 10-1. LUI Command Summary

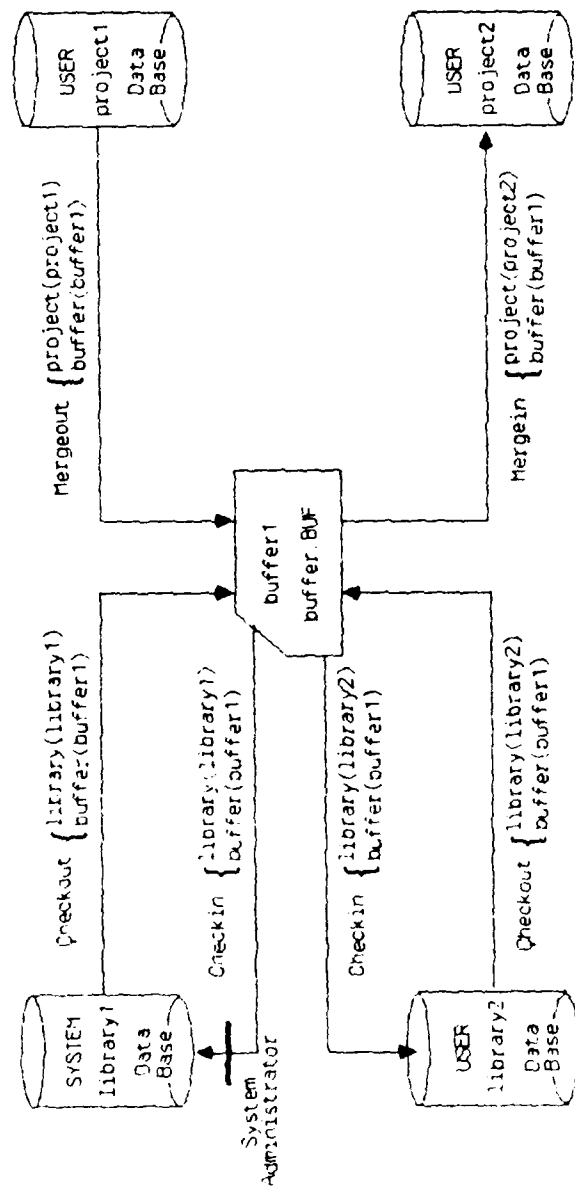


Figure 10-2. Library Utility Data Flow Diagram

10.1 LUI COMMAND: CHECKIN

To move the contents of the buffer to a library for permanent storage, one issues the CHECKIN command. The user is prompted for the name of the model to be checked in, as well as an optional reference number and description.

To enter the CHECKIN sublevel, issue the command:

```
CHECKIN [BUFFER(buffer)] [LIBRARY(library)] [TERM(terminal)]
```

```
CI      [B(buffer)]      [L(library)]      [T(terminal)]
```

where:

[B(buffer)] is an optional parameter indicating the buffer from which entities are to be taken. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY level command.

[L(library)] is a required parameter indicating the library into which the entities are to be entered.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP      - HP2647A or HP2648a terminal  
HP23    - HP2623 terminal  
TEK     - TEK4105 terminal  
VT      - VT100 terminal
```

FUNCTION RESULT:

The system queries the user for a required model name and an optional document reference number and description. After getting this information, the entities in the buffer are put into the library specified under the given model name.

10.2 LJI COMMAND: CHECKOUT

To copy a model stored in a library to a buffer, one enters the CHECKOUT command. At this point the user can obtain a list of the models contained in the library or a list of the given entity types contained in a named model through the LIST command (see section 10.2.5). Models are copied individually through the EXTRACT command which specifies the model to be copied. A HELP command is available. The CHECKOUT sublevel commands are described in detail in sections 10.2.1 through 10.2.5.

To enter the CHECKOUT sublevel, issue the command:

```
CHECKOUT [BUFFER(buffer)] [LIBRARY(library)] [TERM(terminal)]
CO       [B(buffer)]      [L(library)]      [T(terminal)]
```

where:

[B(buffer)] is an optional parameter naming the buffer into which entities are to be placed. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY command.

[L(library)] is a required parameter indicating the library from which the entities are to be taken.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal
```

FUNCTION RESULT:

The model or models specified by the user are written out to the buffer. From the buffer they can be included in a project with the MERGEIN command.

DELETE {model-name}
D

END
E

EXTRACT {model-name}
EXT

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST {model-name}
*

Figure 10-3. Checkout Command Summary

10.2.1 CO COMMAND: DELETE

The DELETE command instructs the system to delete a specified model from a user's library.

COMMAND SYNTAX:

DELETE {model-name}

D

where:

{model-name} is the name of the model to be deleted from the library.

FUNCTION RESULT:

The specified model is deleted from the library. If a user attempts to delete a model from the system library, the following message is displayed: "THIS ACCOUNT IS NOT AUTHORIZED TO MODIFY THE SYSTEM LIBRARY."

CHECKOUT / END

10.2.2 CO COMMAND: END

The END command causes the system to exit the CHECKOUT sublevel and return the user to the LIBRARY READY Level.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

If any models were selected for extraction, the entities are written to a buffer.

The system then returns to the LIBRARY READY level.

10.2.3 CO COMMAND: EXTRACT

The EXTRACT command instructs the system to copy a model from a library into a buffer.

COMMAND SYNTAX:

EXTRACT {model-name}

EXT

where:

{model-name} is the name of the model to be placed in the buffer.

FUNCTION RESULT:

The model specified is copied from the current library into the current buffer.

10.2.4 CO COMMAND: HELP

The HELP command provides the user with access to help information about the current CHECKOUT sublevel and about other aspects of the AISIM system.

COMMAND SYNTAX:

HELP [subtopic],...,[subtopic]

HELP [@topic],[topic-name],[subtopic],...,[subtopic]

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the CHECKOUT function is displayed. The commands acceptable at the current CHECKOUT level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the CHECKOUT help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

10.2.5 CO COMMAND: LIST

The LIST command enables the user to obtain a list of the models contained in a system or user library, or to list the entities in a particular model.

COMMAND SYNTAX:

LIST {model-name}

LIST *

L

where:

{model-name} is the name of a model in the library

* is a literal parameter, indicating all models in the library.

FUNCTION RESULT:

If the parameter {modelname} is used, the system will display a list of the names of the entities in the indicated model. After the names of each entity type are displayed, the user is given the option of continuing to list model entities or of returning to the CHECKOUT ready level. If the parameter * is used, the system will display a list of the names of all the models in the library.

10.3 LUI COMMAND: CONVERT

The CONVERT command enables a user to convert a version 3.0 or 4.0 project database into a 5.0-compatible database. Old databases are incompatible with version 5.0, so all old databases must be converted before they can be used with version 5.0.

COMMAND SYNTAX:

```
CONVERT [PROJECT(project)] [TERM(terminal)] DBVERS(version)
```

```
CONV    [P(project)]      [T(terminal)]    DBV(version)
```

where:

[P(project)] is a required parameter indicating the name of the project being converted.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP      - HP2647A or HP2648A terminal
HP23    - HP2623 terminal
TEK     - TEK4105 terminal
VT      - VT100 terminal
```

DBVERS(version) is a required parameter indicating the version of the data base being converted. The valid versions are V30 and V40.

FUNCTION RESULT:

The project database project.DBF is saved in a database called project.V30 or project.V40 based on its version. The project database is then converted to a version 5.0 database and stored in the project name project.DBF. This database is now suitable for use with all AISIM version 5.0 functions.

10.4 LUI COMMAND: HELP

The HELP command provides the user with access to help information about the current LUI sublevel and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified help, information on the LUI function is displayed. The commands acceptable at the current LUI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the LUI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

10.5 LUI COMMAND: MERGEIN

To move the contents of a buffer to a project database, one enters the MERGEIN command, specifying the name of the buffer and the name of the project into whose database the buffer contents are to be copied.

COMMAND SYNTAX:

```
MERGEIN [PROJECT(project)] [BUFFER(buffer)] [TERM(terminal)]
```

```
MI      [P(project)]      [B(buffer)]      [T(terminal)]
```

where:

[P(project)] is a required parameter indicating the name of the project into which the entities are to be merged.

[B(buffer)] is an optional parameter indicating the name of the buffer in which the entities are stored. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY command.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP    - HP2647A or HP2648A terminal
HP23  - HP2623A terminal
TEK   - TEK4105 terminal
VT    - VT100 terminal
```

FUNCTION RESULT:

If no entity in the buffer is the same as an entity already present in the database, the system responds:

```
0 CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION
```

in which case the copying of the buffer contents will be completed and the user will be returned to the LIBRARY READY level. If one or more names of entities conflict with ones already in the project database, the user will be prompted with:

```
n CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION
```

where "n" is the number of conflicts. The system then asks:

```
DO YOU WISH TO RESOLVE THESE CONFLICTS?
```

Answering "no" aborts the Mergein. If the answer is "yes", the system will then present the name of an entity which stands in conflict.

The user now has three command options to resolve the naming conflict. First, he may command that the entity in the database be deleted in favor of the one of the same name in the buffer. This is done by entering REPLACE (RP). Secondly, he may command that the entity in the buffer which aroused the naming conflict be disregarded in the transferral from the buffer to the database. This is done by issuing the command IGNORE (IG). Thirdly, one may resolve the naming conflict by giving the entity in the buffer a new name. This is done by means of the command RENAME (RN) whose one parameter is the new name the user wishes to give the entity. If the user should select as a new name one that is also being used, the system will respond with a prompt for a different name. These commands are described in detail in sections 10.5.1 through 10.5.6.

This cycle of naming conflict resolution will be repeated until all of the naming conflicts have been resolved. The system will then tell the user that MERGEIN initialization has been completed, do the MERGEIN and automatically return the user to the LIBRARY READY level.

NOTE: Resources associated with an architecture are not subject to the REPLACE command.

END
E

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

IGNORE
IG

INFO

RENAME {name1}
RN

REPLACE
RP

Figure 10-4. Mergein Command Summary

MERGEIN / END

10.5.1 MI COMMAND: END

The END command, issued at the MERGEIN sublevel causes the system to exit the MERGEIN sublevel and returns the user to the LIBRARY READY level.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The system returns to the LIBRARY READY level.

10.5.2 MI COMMAND: HELP

The HELP command provides the user with access to help information about the current MERGEIN sublevel and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the MI function is displayed. The commands acceptable at the current MI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the MI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

10.5.3 MI COMMAND: IGNORE

The IGNORE command enables the user to resolve any naming conflicts encountered at the MERGEIN sublevel in favor of the entities that already exist in the target database.

COMMAND SYNTAX:

IGNORE

IG

FUNCTION RESULT:

The entity indicated by the prompt is not copied into the project database. The system then prompts the user with the next naming conflict, if any, and proceeds with MERGEIN operation.

10.5.4 MI COMMAND: INFO

The INFO command furnishes the user with information on the options available to resolve naming conflicts encountered in the MERGEIN sublevel.

COMMAND SYNTAX:

INFO

IN

FUNCTION RESULT:

The screen displays the following information:

IGNORE: THIS OPTION CAUSES THE NAMED ENTITY IN THE BUFFER TO BE EXCLUDED FROM THE MERGEIN OPERATION

RENAME: THIS OPTION CHANGES ALL OCCURANCES OF THE ENTITY NAME IN THE BUFFER TO THE NAME SPECIFIED BY THE USER

REPLACE: THIS OPTION DELETES THE NAMED ENTITY FROM THE USER DATA BASE, ALLOWING THE ENTITY IN THE BUFFER TO BE MERGED IN

END: THIS OPTION TERMINATES THE MERGEIN PRE-PROCESSING WITHOUT RESOLVING ANY MORE NAMING CONFLICTS AND RETURNS TO THE LUI READY LEVEL

10.5.5 MI COMMAND: RENAME

The RENAME command allows the user to resolve a naming conflict encountered during the MERGEIN operation by giving entities in the buffer a unique name.

COMMAND SYNTAX:

```
RENAME {name1}
```

```
RN
```

where:

{name1} is the new name the entity is to be given.

FUNCTION RESULT:

The system checks to see whether the new name given to the entity creates any naming conflicts. If it does, the system will prompt the user to that effect, and await a new name. If the new name does not create any conflicts, the entity is copied into the project database under its new name. If there are naming conflicts with further entities, the system then prompts the user for their resolution. If there are no remaining naming conflicts, the MERGEIN operation begins.

MERGEIN /REPLACE

10.5.6 MI COMMAND: REPLACE

The REPLACE command enables the user to resolve a naming conflict encountered in the MERGEIN sublevel in favor of entities that exist in the buffer.

COMMAND SYNTAX:

REPLACE

RP

FUNCTION RESULT:

The entity indicated in the prompt is written into the database and the old entity of the same name is deleted. The system then proceeds to consideration of the next naming conflict if any exist. Otherwise, the MERGEIN operation begins.

10.6 LUI COMMAND: MERGEOUT

When the user wishes to place entities from a project database into a buffer, he does so via the MERGEOUT command, specifying the name of the project and the name of the buffer into which the entities are to be copied. Entities in the project are copied one at a time by name through the SELECT command. If the user needs a list of the entities of a given type, he may obtain one through the LIST command. Also available here is the HELP command which provides a menu of the other available commands. The END command will return the user to the LIBRARY READY level. These commands are described in detail in sections 10.6.1 through 10.6.4.

To obtain access to the MERGEOUT sublevel, issue the command,

```

MERGEOUT  [PROJECT(project)] [BUFFER(buffer)] [TERM(terminal)]

MO        [P(project)]      [B(buffer)]      [T(terminal)]

```

where:

[P(project)] is a required parameter indicating the name of the project from which the entities are to be copied.

[B(buffer)] is an optional parameter indicating the name of the buffer into which the entities are to be transferred are stored. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY level command.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```

HP    - HP2647A or HP2648A terminal
HP23  - HP2623 terminal
TEK   - TEK4105 terminal
VT    - VT100 terminal

```

FUNCTION RESULT:

The user is given a "*" prompt, from which he can issue one of the following commands.

- 1) LIST {entity-type}, to list entities in project database.
- 2) SELECT {entity-type},{entity-name}, to select an entity to be merged out of the project database.
- 3) END, which will terminate the selection of entities to be copied.

END
E

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST {entity-type}/SEL
L

SELECT {entity-type},{entity-name}
S

Figure 10-5. Mergeout Command Summary

MERGEOUT / END

10.6.1 MO COMMAND: END

The END command terminates the session at the MERGEOUT sublevel and causes entities in the current project database which have been flagged by the SELECT command to be copied into the current buffer.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The user will be prompted with the question:

DO YOU WANT TO LIST YOUR SELECTIONS ON THE SCREEN?

A "no" answer will cause the Mergeout procedure to take place. When all of the flagged entities have been copied into the buffer the system will return to the LIBRARY READY level.

A "yes" answer will produce a list of the entities flagged in the SELECT command. The user will then be prompted as to whether he wishes to proceed with the Mergeout operation. A "yes" answer to this second question will cause the flagged entities to be copied into the current buffer and the system will return to the LIBRARY READY level. A "no" answer will return the user immediately to the LIBRARY READY level.

10.6.2 MO COMMAND: HELP

The HELP command provides the user with access to help information about the current MERGEOUT sublevel and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the MO function is displayed. The commands acceptable at the current MO level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the MO help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

10.6.3 MO COMMAND: LIST

The LIST command enables the user to obtain a list of the names of the entities of a given type that are contained in the current project.

COMMAND SYNTAX:

```
LIST {entity-type}  
LIST SEL  
L
```

where:

{entity-type} is the type of entity. The valid entity types are the following:

Action	A
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Table	T
Variable	V

SEL indicates that the user wishes to list the entities selected up to that point.

FUNCTION RESULT:

The screen will display a list of the names of the entities of the specified type in the current project or the names of the entities currently selected for mergeout.

10.6.4 MO COMMAND: SELECT

The SELECT command allows the user to specify which entities are to be merged out of a project database to a buffer. Scenarios and Loads cannot be selected.

COMMAND SYNTAX:

```
SELECT {entity-type},{entity-name}  
S
```

where:

{entity-type} is the type of entity to be merged out. The valid entity types are the following:

Action	A
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Table	T
Variable	V

{entity-name} is the name of the entity to be merged out.

FUNCTION RESULT:

The specified entity is flagged for the Mergeout operation. The operation will take place only when the END command is issued.

SECTION 11

HELP EDITOR INTERFACE

At the HEI level of operation, and its lower level, commands are available to request information about the use of various AISIM functions and information pertinent to various AISIM concepts (HELP) and to generate user supplied help information (UPDATE). The UPD sublevel of the HEI is invoked to create and edit user provided information. This information may cover a wide range of topics but will fall under three basic headings: notes, guidelines, and procedures. This instructional information can later be accessed by all AISIM users by means of subsequent invocations of the HELP command.

These commands are summarized in the command summary in figure 11-1 and are described on the pages that follow.

11.1 HEI COMMAND SUMMARY

END

HELP [subtopic],..., [subtopic]

HELP [@topic], [topic-name], [subtopic], ..., [subtopic]

UPDATE

U

Figure 11-1. HEI Command Summary

HEI / END

11.1.1 HEI COMMAND: END

The END command is used to terminate a HEI session

COMMAND SYNTAX:

END

FUNCTION RESULT:

The HEI session is ended, the system will return the AISIM READY level and the screen will display

AISIM READY>

11.1.2 HEI COMMAND: HELP

The **HELP** command provides the user with access to help information about the current HEI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

HELP [subtopic],..., [subtopic]

HELP [@topic], [topic-name], [subtopic],..., [subtopic]

where:

[subtopic] is an optional parameter indicating the name of a subtopic, at which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level of the help system is the current function, and should be displayed. Available HELP topics are:

Topic	Acceptable Abbreviation
FUNCTION LEVEL	F
ACCEPT	A
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the HIL function is displayed. The commands acceptable at the current HIL level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the HEI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "4" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

11.1.3 HEI COMMAND: UPDATE

The UPDATE command is used to create an informational help message.

COMMAND SYNTAX:

UPDATE

U

FUNCTION RESULT:

The UPD level of operation is entered. This operation allows the user to manage user-supplied instruction. A # prompt is provided for the user to input UPD commands. These commands are discussed in section 11.2.

11.2 UPDATE (UPD)

The UPD capability augments the current comprehensive multilevel HELP facility by providing the user the ability to provide user specific information. This information can potentially cover a wide range of topics under three basic headings: notes, guidelines, and procedures.

All instruction supplied by the user will be categorized under one of the above three topics. The latter two may be used to provide information that can be used to manage and guide the use of AISIM while the notes category can contain miscellaneous information. The actual decision as to how to decompose the user-supplied instructional information into these three areas is, of course, completely up to the discretion of the user supplying the information.

When creating and editing help topics in the UPD sublevel, the system prompts the user for information by the use of forms as shown in section 4.2. The forms editor is the one described in section 6 for the DUI.

While the user is in the UPD, all changes are made to a working copy of the help database. When the user issues a SAVE command during or at the conclusion of the session, the working database is copied to the help database. To avoid potential conflicts with updating the help database, only one AISIM user is allowed to enter the UPD level at a time. The working database is created on the current users id and is a fairly large database file.

Note: If error CPYDB01,Flag=38 occurs while entering UPDATE there is a good possibility that disk quota on the current id is not sufficient to create the working copy of the help database.

The UPD level of commands are described in detail on the following pages.

11.2.1 UPDATE COMMAND SUMMARY

ADD {topic},{topic-name}

CHANGE {topic},{topic-name}
CHG

DELETE {topic},{topic-name}
DEL

END

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST {topic}
L

SAVE
S

Figure 11-2. UPDATE Command Summary

11.2.2 UPDATE COMMAND: ADD

The ADD command is used to create an informational help message.

COMMAND SYNTAX:

ADD [topic],[topic-name]

where:

[topic] is a required parameter indicating any valid user-supplied information topic type.

Topic may be any of the following:

topic	Acceptable Abbreviation
PROCEDURE	P
GUIDELINE	G
NOTE	N

[topic-name] is a required parameter specifying the name of the new help information message to be created.

If [topic] or [topic-name] is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the UPD Ready state - # prompt.

FUNCTION RESULT:

The ADD command will create an information message of the specified type. A form will be displayed for the user to enter the information.

Up to 20 lines of information may be entered. If less than 20 lines of text is entered the informational message will be assumed complete upon the occurrence of two blank lines. This will eliminate the display of multiple blank lines when the message is later accessed for viewing.

11.2.3 UPDATE COMMAND: CHANGE

The CHANGE command is used to modify an informational help message.

COMMAND SYNTAX:

CHANGE {topic},{topic-name}

CHG

where:

{topic} is required parameter indicating any valid user-supplied information topic type.

Topic may be any of the following:

topic	Acceptable Abbreviation
PROCEDURE	P
GUIDELINE	G
NOTE	N

{topic-name} is a required parameter specifying the name of the help information message to be modified.

If {topic} or {topic-name} is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the UPD Ready state - # prompt.

FUNCTION RESULT:

The CHANGE command will display the information message specified on a form that will allow modification of its contents.

11.2.4 UPDATE COMMAND: DELETE

The DELETE command is used to remove an informational help message.

COMMAND SYNTAX:

```
DELETE {topic},{topic-name}
```

```
DELETE {topic},{topic-name},...,{topic-name}
```

```
DEL
```

where:

{topic} is a required parameter indicating any valid user supplied information topic type.

Topic may be any of the following:

topic	Acceptable Abbreviation
PROCEDURE	P
GUIDELINE	G
NOTE	N

{topic-name} is a required parameter specifying the name of the help information message to be modified.

If {topic} or {topic-name} is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the UPD Ready state - # prompt.

FUNCTION RESULT:

The DELETE command will remove the information specified from the available user-supplied help messages.

UPDATE / END

11.2.5 UPDATE COMMAND: END

The END command terminates an UPDATE session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The UPDATE session is ended. The working database is closed. If a SAVE command has not been given since the last ADD, DELETE or CHANGE command, the user is asked if the working database is to be saved. The query is :

SAVE (Y/N)?

If the user answers "Y", the working database is saved into the real database and the session is ended. Control is passed to the HEI level (level 4). If the user answers "N", the session is ended and the working database is not saved. Control is passed to the HEI level. Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the UPD level - * prompt.

11.2.6 UPDATE COMMAND: HELP

The HELP command provides the user with access to help information about the current UPD interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

```
HELP [subtopic],...,[subtopic]
```

```
HELP [@topic],[topic-name],[subtopic],...,[subtopic]
```

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTIONAL LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic-name] is an optional parameter indicating the name for the new topic level topic.

FUNCTION RESULT:

If no path is specified, help information on the UPD function is displayed. The commands acceptable at the current UPD level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the UPD help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to seek additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "0" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

11.2.7 UPDATE COMMAND: LIST

The LIST command is used to list the informational help messages.

COMMAND SYNTAX:

LIST {topic}

L

where:

{topic} is a required parameter indicating any valid user-supplied name or topic type.

Topic may be any of the following:

topic	Acceptable Abbreviations
PROCEDURE	P
GUIDELINE	G
NOTE	N

If {topic} is missing or invalid, the user is prompted for a valid command.

A carriage return in response to the prompt aborts the command. The user is returned to the UPD Ready state - # prompt.

FUNCTION RESULT:

The LIST command will display the names of the current information messages of the specified type.

UPDATE / SAVE

HELP - PLATE (MMAC) - 3

The SAVE button saves the current UPDATE session.

MMAC - 3

SAVE

MMAC - 3

The SAVE button saves the help information created or updated by the user into the database of help messages.

740-A190 171

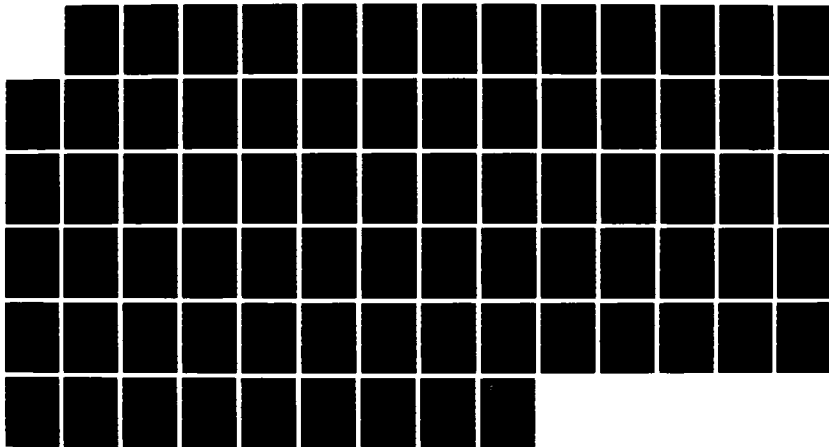
AUTOMATED INTERACTIVE SIMULATION MODEL (AISM) VAX
VERSION 30 USER'S MANU. (U) HUGHES AIRCRAFT CO
FULLERTON CA GROUND SYSTEMS GROUP V ALLERTON ET AL.

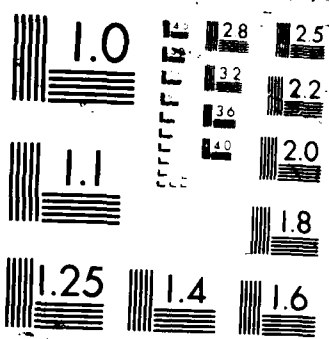
4/4

UNCLASSIFIED

29 MAY 87 ESD-TR-87-220 F19620-06-C-0070 F/B 12/3

NL





SECTION 12

FILE MANAGEMENT USER INTERFACE (FUI)

The FUI is used to create or edit a file that is to be read from by a READ Primitive during a simulation run. The FUI has commands to enable the user to add and delete lines from the file by specifying line numbers upon which the commands are to operate. The FUI can operate in an error checking or non-error checking mode. In error checking mode the user specifies a project data base that will be accessed during the FUI session. The user will be able to list the names of entities in the data base, and whenever the user places the name of an entity into the file, the existence of the entity will be verified in the data base. In non-error checking mode, the only verification that takes place is that numeric values do not contain any characters, and alpha literals start with a dollar sign.

Each value placed in a file is added on a separate line. Files not created under the FUI can be edited under the FUI as long as each value is on a separate line. However, the simulation does not require that values be on separate lines. The following types of data can be placed in a READ file:

- Process name
- Resource name
- Item name
- Queue name
- Table name
- Action name
- Alpha literal
- Numeric

The FUI commands to manage the READ file are illustrated in figure 12-1 and described on the pages that follow it.

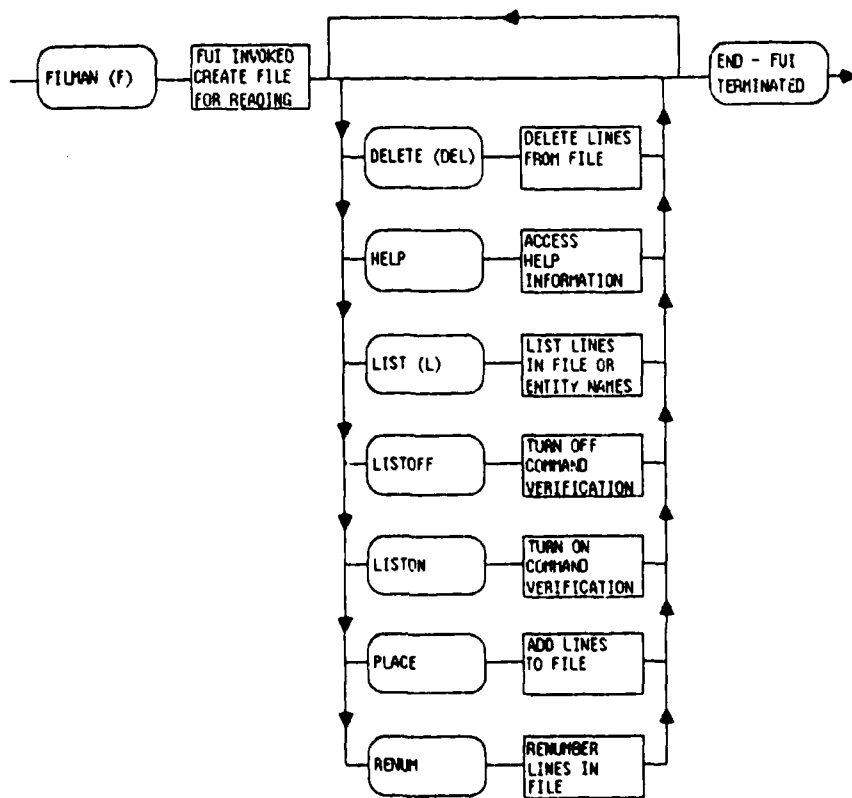


Figure 12-1. File Management User Interface Commands

12.1 FUI COMMAND SUMMARY

Figure 12-2 contains a summary of the FUI level commands.

```
DELETE {line-number1},{line-number2}
DEL

END

HELP [subtopic],...,[subtopic]
HELP [@topic],[topic-name],[subtopic],...,[subtopic]

LIST [line-number1],[line-number2]
LIST [entity-type]
L

LISTOFF

LISTON

PLACE {type},{name},{line-number}
P

RENUM
```

Figure 12-2. FUI Command Summary

12.1.1 FUI COMMAND: DELETE

The DELETE command is used to remove lines from the file.

COMMAND SYNTAX:

DELETE {line-number1},{line-number2}

DEL

where:

{line-number1} is a required parameter indicating the first line to be deleted.

[line-number2] is an option parameter indicating the last line for a range of lines to be deleted.

FUNCTION RESULT:

If the user is in LISTON mode (see section 12.1.6) the deleted lines are listed. In either mode, a message indicating the number of lines that were deleted is displayed.

FUI / END

12.1.2 FUI COMMAND: END

The END command is used to terminate an FUI session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

If the user has made any changes to the data file, AISIM queries the user as to whether the changes should be saved. If the user responds "yes", the temporary file is copied into the permanent data file. If the user responds "no", the permanent data file remains unchanged. The user is then returned to the AISIM READY level.

12.1.3 FUI COMMAND: HELP

The HELP command provides the user with access to help information about the current FUI interface and about other aspects of the AISIM system.

COMMAND SYNTAX:

HELP [subtopic],...,[subtopic]

HELP [@topic],[topic~name],[subtopic],...,[subtopic]

where:

[subtopic] is an optional parameter indicating the name of a subtopic about which the user would like information. Successive subtopics contain more detailed information.

[@topic] is an optional parameter indicating that the top level topic should not be the current function but should be as specified. Available HELP topics are:

topic	Acceptable Abbreviation
FUNCTION LEVEL	F
CONCEPT	C
GUIDELINE	G
PROCEDURE	P
NOTE	N

[topic~name] is an optional parameter indicating the name for the new top level topic.

FUNCTION RESULT:

If no path is specified, help information on the FUI function is displayed. The commands acceptable at the current FUI level of operation are listed as subtopics indicating that further help is available on them. HELP with no path specified displays the FUI help message text and available subtopics, followed by the prompt:

Subtopic?

If you type in a subtopic name, a HELP message on the subtopic will be displayed. If one or more subtopics exist at this level, HELP will prompt you for another subtopic allowing you to see additional help information. Pressing a <cr> will terminate your descent for additional help.

If you know precisely what information you need, you can access it directly by including a path parameter which specifies the subtopics to move down through to locate the help message. Each subtopic listed in the path must be separated from the previous by a comma.

After you have located the help information you wanted and prior to terminating your initial request for help, the system will prompt you for another topic. The help topic initially is the AISIM function you invoked help from. You can continue to view information on this topic-name by simply entering another subtopic path. Information on other AISIM functions, AISIM modeling concepts, and user-supplied instruction can be obtained by changing the help topic. You can change the help topic by typing the special character "@" followed by the new top level HELP topic and the topic-name. Once again you can specify a subtopic path to the information you want. If you are not aware of the topic-names under a specific topic, entering the topic parameter will display a message listing the possible topic names.

12.1.4 FUI COMMAND: LIST

The LIST command is used to display lines in the file or the names of entities in an associated error checking data base.

COMMAND SYNTAX:

LIST [line-number1],[line-number2]

LIST [entity-type]

L

where:

[line-number1] is an optional parameter indicating the first, or only, line to be listed.

[line-number2] is an optional parameter indicating the last number in a range of numbers to be listed.

[entity-type] is an optional parameter indicating the type of entities whose names are to be listed.

FUNCTION RESULT:

If line numbers are specified, then lines from the file are listed. If "entity-type" is specified and there is an error checking data base, then the names of entities of the specified type are listed.

12.1.5 FUI COMMAND: LISTOFF

The LISTOFF command is used to turn off verification for the FUI commands.

COMMAND SYNTAX:

LISTOFF

FUNCTION RESULT:

In LISTOFF mode, the results of PLACE and DELETE commands are not reflected in the screen display.

12.1.6 FUI COMMAND: LISTON

The LISTON command is used to turn on verification for the FUI commands.

COMMAND SYNTAX:

LISTON

FUNCTION RESULT:

When LISTON is set, the results of PLACE and DELETE commands are verified on the screen.

12.1.7 FUI COMMAND: PLACE

The PLACE command is used to add data to the file.

COMMAND SYNTAX:

```
PLACE {type},{name},{line-number}
```

P

where:

{type} is a required parameter indicating the type of data being added to the file. The valid types and their abbreviations are as follows:

Process	P
Resource	R
Item	I
Process[]	P[]
Resource[]	R[]
Item[]	I[]
Queue	Q
Table	T
Action	A
Alpha	AL
Numeric	N

[name] is an optional parameter indicating the name of an entity whose attributes are to be placed in the file.

{line-number} is a required parameter indicating the line number at which the data is to be added to the file.

FUNCTION RESULT:

A form is displayed for the user to enter the data that is to be placed in the file. If "type" is an entity, a form prompting for the name of the entity is displayed. If error checking is in effect, the existence of the entity is verified in the data base. If "type" is alpha or numeric, a form prompting for the data is displayed, and verification is made as to whether the data is an alpha literal or numeric. If type is "entity[]", a form prompting for the attributes of the entity is displayed. If error checking is not in effect, the form will contain spaces for up to fifteen attributes. If error checking is in effect, the names of attributes of the named entity are displayed next to the form fields so the user will know what attributes are required.

When the form is entered, the data is added to the file.

12.1.8 FUI COMMAND: RENUM

The RENUM command is used to renumber the data file.

COMMAND SYNTAX:

RENUM

FUNCTION RESULT:

The lines in the data file are renumbered by whole numbers. When the user inserts data between two lines that are sequential (such as 3 and 4), the increment is tenths (i.e. 3.1, 3.2...). If necessary the user can use hundredths to place data between two lines such as 3.2 and 3.3. The RENUM command will go through the file and renumber all the lines by whole numbers.

SECTION 13

AISIM SIMULATION REPORTS

When a simulation is run, a number of Processes are initiated at various times throughout the simulation period. As their execution proceeds they contend for available Resources such as machines and operators. The simulation stops at the end of a predefined period and produces output statistics.

In general, any high-level performance factor measurable on a real system in terms of time, percentages, or counts of events can be measured during the model run. Experiments that are virtually impossible to run on a real system can be constructed and easily measured in the model. Specifically, measures that may be obtained are:

- Resource utilization statistics
- Total number of Processes completed
- Average elapsed time for Process completion
- System and job delays associated with actions
- Statistics on queue sizes and timing
- Variable changes during simulation
- System and job delays associated with Resources
- Execution count of Process steps

Two forms of statistical output are available to the user as a result of the simulation. Interactive output, displayed on the terminal screen, is available at any user-defined breakpoint, at the end of simulation periods, or at the end of the simulation.

The second form of output is a listing, obtained off-line, which lists the simulation measures mentioned above.

The following sections describe the simulation outputs and how to obtain them.

13.1 INTERACTIVE RESULTS AND HOW TO OBTAIN THEM

Interactive results can be viewed on the terminal while in the AUI level. A review of the AUI level shows that several commands are available for viewing data after simulation periods, after breakpoints, and after simulation termination. The DEFLOT command is used before simulation is started to select the graphs that the user wishes to view after simulation (see the DEFLOT command description in section 7.2 for attributes and statistics of entities that can be graphed). The LISTVAL command can be used at the points mentioned above to view simulation data concerning model entities (see the LISTVAL command description in section 7.11 for attributes and statistics of entities that can be viewed). The PLOT

command is also used at the points mentioned above to view graphically the statistics which were kept due to the DEF PLOT plot definitions. See the PLOT command definition in section 7.12 for examples of the forms and graphs that are displayed to the user as a result of this command.

13.2 REPORT RESULTS AND HOW TO OBTAIN THEM

The commands to view and print results are available at the AISIM READY level. As the simulation executes, simulation results are automatically stored in a database file named:

project.RPT

where:

project indicates that the model output report to be accessed was generated by an analyze session on the design database named PROJECT.

Two AISIM READY level commands are available to manipulate this data file. The PRINT command (see section 5.20) is used to print a listing of the simulation report at the local hardcopy facility. The EDIT command (see section 5.7) allows the user to view the project.RPT file through the use of the EDT text editor. See section 13.3 for a brief discussion of relevant EDT text editor commands. See the EDT Users Manual for additional information on the EDT text editor.

The project.RPT file contains a number of reports that describe the model that was simulated and the results of the simulation. On the following pages each of these reports is described and examples of results are given.

INITIALIZATION REPORT: This report displays the time at which the model was run, any descriptive text entered by the user, and the contents of the model inputs as used during this simulation. Elements of this report are:

- 1) Global Constant Definition
- 2) File Definition
- 3) Table Definition
- 4) Global Variable Definition
- 5) Item Definition
- 6) Queue Definition
- 7) Resource Definition
- 8) Architecture Legal Path Definition
- 9) Action Definition
- 10) Process Definition
- 11) Load Definition
- 12) Scenario Definition

Figures 13-1 through 13-5 show the various parts of a typical initialization report.

```

#####
S I M U L A T I O N   R E P O R T
S           AISIM VERSION 5.0           S
S           HUGHES AIRCRAFT COMPANY      S
S           05/15/87                     S
#####

```

04/30/1987 10:29:55

TESTDBB

TEST KEYWORDS, PARAMETER PASSING AND READ/WRITE

GLOBAL CONSTANT DEFINITION.....

CONSTANT	INITIAL	
MNEMONIC	VALUE	COMMENT
=====	=====	=====

FILE DEFINITION.....

FILE		
MNEMONIC		COMMENT
=====	=====	=====
FILE1	FILE TO READ FROM	
FILE2	FILE TO WRITE TO	

TABLE DEFINITION....

GLOBAL VARIABLE DEFINITION.....

VARIABLE	INITIAL	
MNEMONIC	VALUE	COMMENT
=====	=====	=====
V_CHNL	0	GLOBAL VARIABLE TO HOLD CHANNEL
V_CLOCK1	0	GLOBAL VARIABLE OF CLOCK FIRST SAMPLE
V_CLOCK2	0	GLOBAL VARIABLE OF CLOCK SECOND SAMPLE
V_CNODE	A	GLOBAL VARIABLE OF CURRENT NODE INITIALIZED TO RES
V_NXTND	B	GLOBAL VARIABLE INITIALIZED TO RESOURCE

Figure 13-1. Initialization Report - Constants, Files, Tables, and Global Variables

ITEM DEFINITION.....

ITEM	DESCRIPTION	
ACK	ACKNOWLEDGEMENT GENERATED AT COMM CENTERS BOUND FOR S	
ATTR.	INITIAL	
NAME	VALUE	
HOPS	1	
LENGTH	ACKLENGTH	
MESS	0	
ORIGIN	0	
RETRAN	1	
TNODE	0	
TYPE	\$GOOD	

ITEM	DESCRIPTION	
MSG04	MESSAGES GENERATED AT S-NODES BOUND FOR COMM CENTERS	
ATTR.	INITIAL	
NAME	VALUE	
ACKREC	0	
DEST1	1	
DEST2	1	
DEST3	1	
DEST4	1	
ENDTM	99999999	
ERRPROB	ERRPRB04	
HOPS	HOPS04	
LENGTH	LENGTH04	
NXTACKNM	1	
ORIGIN	S04	
RETRAN	1	
SATDELAY	DELAY04	
SNUM	S04NUM	
STARTTM	\$CLOCK	

QUEUE DEFINITION.....

QUEUE	MAXIMUM	
MNEMONIC	SIZE	COMMENT
BUFFER	INFINITE	BUFFER FOR MESSAGES AWAITING PROCESSING

Figure 13-2. Initialization Report - Items and Queues

RESOURCE DEFINITION.....

RESOURCE MNEMONIC	TOTAL # UNITS	INITIAL # UNITS	DESCRIPTION
B1	1	1	RESOURCE FOR CPU NODE
=====			
ATTR.	NAME	INITIAL VALUE	
=====			
D_CS		0	
M_CS		2	
M_ROUTE		0	
RATE		SECS_CHR	
=====			
B1B2	1	1	RESOURCE FOR CHANNEL CONNECTOR
=====			
ATTR.	NAME	INITIAL VALUE	
=====			
RATE		56	
=====			
B2	1	1	RESOURCE FOR CPU NODE
=====			
ATTR.	NAME	INITIAL VALUE	
=====			
D_CS		0	
M_CS		2	
M_ROUTE		0	
RATE		SECS_CHR	

ARCHITECTURE LEGAL PATH DEFINITION

FROM DEVICE	TO DEVICE	NEXT DEVICE	VIA LINK
=====	=====	=====	=====
B1	B2	B2	B1B2
B1	B3	B2	B1B2
B1	B4	B2	B1B2
B1	B5	B2	B1B2
B1	B6	B2	B1B2
B1	H1	H1	H81_B
B1	H2	B2	B1B2
B1	H3	B2	B1B2
B1	H4	B2	B1B2
B1	H5	B2	B1B2
B1	H6	B2	B1B2
B2	B1	B3	B2B3
B2	B3	B3	B2B3
B2	B4	B3	B2B3
B2	B5	B3	B2B3
B2	B6	B3	B2B3
B2	H1	B3	B2B3
B2	H2	H2	H82_B
B2	H3	B3	B2B3
B2	H4	B3	B2B3
B2	H5	B3	B2B3
B2	H6	B3	B2B3

Figure 13-3. Initialization Report - Resources and Architecture
Legal Path Table

ACTION DEFINITION.....

ACTION	
MNEMONIC	COMMENT
=====	=====
READ MSG	DELAY AT RECEIVE TO PROCESS MESSAGE
SENDING	DELAY AT TRANSMIT TO DELIVER MESSAGE TO BUFFER

PROCESS DEFINITION.....

PROCESS	
MNEMONIC	DESCRIPTION
=====	=====
RECEIVE	RECEIVE MESSAGES FROM TRANSMIT

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
=====	=====	=====	=====	=====	=====
	START	STATION2	NO		
	TEST	SEMA	ABORT		TEST FOR BUFFER USE
	REMOVE	FIRST	MSG	BUFFER	REMOVE BY FIFO DISCIPLINE
	COMPARE	MSG		EQ	WHEN MSG=0 BUFFER IS EMPTY
		0		ABORT	
	ASSIGN	MSG	LENGTH		MESSAGE LENGTH IS READ
		ALPHA			
	EVAL	MU			CALCULATE RECEPTION TIME
		ALPHA*GAMMA2			
	READ_MSG	CONSTANT	MU		TIME TO PROCESS MESSAGE
		SECONDS	RESUME		
	DESTROY	MSG			MSG ELIMINATED FROM SYSTEM
ABORT	ENTRY				ENTER FROM COMPARE & TEST
	END				

LOCAL VARIABLES OF PROCESS RECEIVE

=====	=====	=====	=====	=====	=====
1 SEMA	(R)	2 MSG	(I)	3 BUFFER	(Q)
5 MU		6 READ_MSG	(A)		

GLOBAL VARIABLES OF PROCESS RECEIVE

=====	=====
1 GAMMA2	
PROCESS	
MNEMONIC	DESCRIPTION
=====	=====
TRANSMIT	TRANSMITTING MESSAGES TO RECEIVER

ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
=====	=====	=====	=====	=====	=====
	START	STATION1	NO		
	ALLOC	SEMA	1	ALL	RESOURCE FOR SENDING MSG
		\$PRIORITY			
	CREATE	MSG			INTRODUCE MSG INTO SYSTEM
	EVAL	ALPHA			GENERATE RANDOM NUMBER
		RANDOM			
	EVAL	ALPHA			AVERAGE TIME ALPHA
		ALPHA*GAMMA1			
	ASSIGN	ALPHA			SET MESSAGE LENGTH
		MSG	LENGTH		
	EVAL	MU			CALCULATE TRANSMIT TIME
		ALPHA*GAMMA2			
	SENDING	CONSTANT	MU		TIME CONSUMED TRANSMITTING
		SECONDS	RESUME		
	FILE	MSG	LAST	BUFFER	STORE MSG ON BUFFER
	DEALLOC	SEMA	1		RELEASE RESOURCE SEMA
	END				

LOCAL VARIABLES OF PROCESS TRANSMIT

=====	=====	=====	=====	=====	=====
1 BUF1	(R)	2 MSG	(I)	3 ALPHA	
5 SENDING	(A)	6 BUFFER	(Q)		4 MU

GLOBAL VARIABLES OF PROCESS TRANSMIT

=====	=====
1 GAMMA1	2 GAMMA2

Figure 13-4. Initialization Report - Actions and Processes

```

LOAD
MNEMONIC      DESCRIPTION
=====
LOADH6        THIS IS THE LOAD FOR HOST 6
               LOAD      NODES
               =====
               H6

PROCESS
MNEMONIC  MAX #  SCHEDULE  MEAN  DELTA  UNITS  PRIORITY
=====
TOHOST1   20    EXPONENT  1370   0     MSECONDS 0
TOHOST2   20    EXPONENT  1370   0     MSECONDS 0
TOHOST3   20    EXPONENT  1370   0     MSECONDS 0
TOHOST4   20    EXPONENT  1370   0     MSECONDS 0
TOHOST5   20    EXPONENT  1370   0     MSECONDS 0

SCENARIO DEFINITION....

SCENARIO
MNEMONIC      DESCRIPTION
=====
PSCEN

PERIOD  PERIOD  OUTPUT
LENGTH  UNITS   UNITS
=====
15000   MSECONDS MSECONDS

PERIOD  PERIOD  PERIOD  PERIOD  PERIOD  PERIOD  PERIOD
MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC
=====
PER1

TRIGGER  TIME TO SCHEDULE SCHEDULE TRIGGER  TIME TO SCHEDULE SCHEDULE
MNEMONIC SCHEDULE UNITS  PRIORITY MNEMONIC SCHEDULE UNITS  PRIORITY
=====
LOADH1   0     MSECONDS 0          LOADH2   0     MSECONDS 0
LOADH3   0     MSECONDS 0          LOADH4   0     MSECONDS 0
LOADH5   0     MSECONDS 0          LOADH6   0     MSECONDS 0

#### 0 ERRORS WERE DETECTED DURING MODEL INITIALIZATION

```

Figure 13-5. Initialization Report - Loads and Scenario

13.2.1 Constant Report

This report shows the value of the constants at simulation termination. An example of this report is shown in figure 13-6 where the labeled columns have the following significance.

CONSTANT: The name of the Constant

CURRENT VALUE: The Constant's value (in real numbers) at the end of the simulation.

```
SIMULATION TIME =      3241.000000 SECONDS
CONSTANT REPORT

      CURRENT
CONSTANT VALUE...
=====
BER      .00001
CCCHRATE 2.4
CHRATE01 1.2
CHRATE02 2.4
CHRATE03 2.4
CHRATE04 .6
CHRATE05 .075
CHRATE06 .075
CHRATE07 1.2
CHRATE08 .3
CHRATE09 2.4
CHRATE10 .3
CHRATE11 1.2
CHRATE12 .3
CLOCKVAL 0
DELAY01 0
DELAY02 0
DELAY03 0
DELAY04 0
DELAY05 0
DELAY06 0
```

Figure 13-6. Constant Report

13.2.2 Variable Report

Variable reports are divided into the numeric and the non-numeric variables. A sample of the report for numerical variables is shown in figure 13-7, where the columns have the following significance.

VARIABLE: The name of the Variable.

TOTAL SAMPLES: The number of times the Variable has been set to a value over the simulation period, including its initialization at the start of the simulation.

CURRENT: The value of the Variable at the end of the simulation.

MEAN: The mean of all values (including its initial value) that the Variable was set to over the simulation (i.e., the sum of the values divided by TOTAL SAMPLES).

STD DEV: The standard deviation of the values that the Variable was set to over the simulation.

MINIMUM: The minimum value that the Variable took on during the simulation.

MAXIMUM: The maximum value that the Variable took on during the simulation.

SIMULATION TIME = 10000.00000 SECONDS

VARIABLE REPORT

NUMERIC VARIABLES...

VARIABLE	TOTAL SAMPLES	CURRENT...	MEAN....	STD DEV...	MINIMUM...	MAXIMUM...
VABS	2	10.000	10.000	0.	10.000	10.000
VADD	2	10.000	10.000	0.	10.000	10.000
VARCOS	89	2.000	2.000	0.	2.000	2.000
VARCSIN	89	1.142	1.142	+4.012E-08	1.142	1.142
VARCTAN	89	-1.142	-1.142	+2.764E-07	-1.142	-1.142
VBETA	89	0.828	0.777	0.990	0.427	10.000
VBIN	89	100.000	98.882	10.488	0.500	100.000
VCOMP1	2	12.000	6.000	6.000	0.	12.000
VCOMP2	2	53.150	26.575	26.575	0.	53.150
VCOMP3	2	9.000	4.500	4.500	0.	9.000
VCOMP4	2	7.135	3.567	3.567	0.	7.135
VCDS	89	-0.416	-0.416	+1.834E-08	-0.416	-0.416
VCTABLE	8	85.000	50.000	32.977	0.	100.000
VDIV	2	10.000	10.000	0.	10.000	10.000
VDTABLE	8	80.000	47.500	33.072	0.	100.000
VERLANG	89	9.710	11.173	10.250	3.937	100.000
VEXP	89	3.343	8.847	8.990	0.058	53.791
VEXP10	89	100.000	100.000	0.	100.000	100.000
VEXPE	89	100.000	100.000	+1.368E-05	100.000	100.000

Figure 13-7. Numeric Variable Report

The report for Variables taking non-numeric values is illustrated in figure 13-8 where the labeled columns have the following significance.

VARIABLE: The name of the Variable.

CURRENT TYPE: The type of entity or construct that the Variable is set to at the end of the simulation.

CURRENT VALUE: The name of the entity or construct to which the Variable is set at the end of the simulation.

NON-NUMERIC VARIABLES...

VARIABLE	CURRENT TYPE	CURRENT VALUE
=====	=====	=====
ACKST011	ALPHA	\$CORRECT
STATE013	ALPHA	\$CORRECT
STATE014	ALPHA	\$CORRECT
STATE021	ALPHA	\$CORRECT
STATE022	ALPHA	\$ERROR
STATE023	ALPHA	\$CORRECT
STATE024	ALPHA	\$CORRECT
STATE031	ALPHA	\$ERROR
STATE032	ALPHA	\$CORRECT
STATE033	ALPHA	\$CORRECT
STATE034	ALPHA	\$CORRECT
STATE041	ALPHA	\$CORRECT
STATE042	ALPHA	\$CORRECT
STATE043	ALPHA	\$CORRECT
STATE044	ALPHA	\$CORRECT
STATE051	ALPHA	\$ERROR
STATE052	ALPHA	\$ERROR
STATE053	ALPHA	\$ERROR
STATE054	ALPHA	\$ERROR
STATE061	ALPHA	\$ERROR
STATE062	ALPHA	\$ERROR
STATE063	ALPHA	\$ERROR
STATE064	ALPHA	\$ERROR
STATE071	ALPHA	\$CORRECT
STATE072	ALPHA	\$CORRECT
STATE073	ALPHA	\$CORRECT
STATE074	ALPHA	\$CORRECT
VAR	RESOURCE	CPU

Figure 13-8. Non-numeric Variable Report

13.2.3 Item Report

Figure 13-9 illustrates the Item Report, where the labeled columns have the following significance.

ITEM NAME: The name of the Item.

NUMBER CREATED: The number of instances of this Item that have been created with the CREATE or SEND Primitives over the simulation.

NUMBER DESTR'D: The number of instances of this Item that have been destroyed with the DESTROY Primitive over the simulation.

TIME IN SYSTEM - MINIMUM: The minimum time any instance of the Item was in the system.

TIME IN SYSTEM - MAXIMUM: The maximum time any instance of the Item was in the system.

TIME IN SYSTEM - AVERAGE: The average time any instance of the Item was in the system.

TIME IN SYSTEM - STD DEV: The standard deviation in the times the Item spent in the system.

MINIMUM, MAXIMUM, AVERAGE, STD DEV are based on the individual Item instances' time in the system. This statistic is calculated whenever an Item instance is destroyed (with the DESTROY Primitive) and is equal to the time of destruction minus the time of creation (with the CREATE or SEND Primitive). Therefore, Items in the system that have not been destroyed at simulation end will not be reflected in these statistics.

SIMULATION TIME = 100.00000 SECONDS

ITEM REPORT

ITEM NAME	NUMBER CREATED	NUMBER DESTR'D	TIME IN SYSTEM			
			MINIMUM...	MAXIMUM...	AVERAGE...	STD DEV...
=====	=====	=====	=====	=====	=====	=====
AITEM1	10	9	10.00	90.00	50.00	25.82
AITEM2	10	9	10.00	90.00	50.00	25.82
AITEM3	10	9	10.00	90.00	50.00	25.82
AITEM4	9	9	0.	0.	0.	0.
ITEM1	10	10	5.00	5.00	5.00	0.
ITEM2	10	10	4.00	4.00	4.00	0.
ITEM3	10	10	3.00	3.00	3.00	0.
ITEM4	10	10	2.00	2.00	2.00	0.
ITEM5	10	10	1.00	1.00	1.00	0.
ITEM6	10	10	0.	0.	0.	0.

Figure 13-9. Item Report

13.2.4 Resource Report

This report gives statistics on each Resource's presence in the idle state, busy queue, hold queue, and inactive state as well as the number of Processes put into a wait queue for the Resource. These queues are discussed in detail in the section on system defined queues (see section 3.5). Four kinds of statistics are kept on the busy and wait queues: (1) entities put into the queue (INTO), (2) entities taken out of a queue (OUT OF), (3) the number in the queue (#), and (4) the time entities spent in the queue (TIME). Statistics on the number in the state are kept for the idle and inactive states. Total samples and time in queue statistics are tabulated for the hold queue.

An example of the Resource Report on these states and queues is shown in figure 13-10. For each row of each queue or state the numbers have the following significance.

The TOTAL NUMBER of the INTO and OUT OF rows indicate the number of entities that were, respectively, placed in or taken out of the queue.

The TOTAL NUMBER for HOLD TIME statistics indicates the number of samples in the time-in-hold-queue statistics.

The CURRENT # is the number of entities in the queue or state at the time the simulation run was completed.

The MEAN # is the time weighted average of the number of entities in the queue or state over the simulation.

The STD DEV # is the standard deviation in the number of entities in the queue or state over the simulation.

The MINIMUM # is the minimum number of entities in the queue or state at one time over the simulation.

The MAXIMUM # is the maximum number of entities in the queue or state at one time over the simulation.

The MEAN TIME is the average time entities spent on the queue.

The STD DEV TIME is the standard deviation in the time that the entities spent on queue.

The MINIMUM TIME is the minimum time any entity was in the queue.

The MAXIMUM TIME is the maximum time any entity was in the queue.

The REQUEST TIME statistics provide the mean, standard deviation, minimum and maximum of the time it took for the request for each unit of the Resource to be satisfied. I.e., the request time is the difference between the time an allocate request is made and the time the Resource unit is placed in the busy queue.

The HOLD TIME statistic provides the mean, standard deviation, minimum and maximum time that Resource units were allocated but waiting for the request, of which they were a part, to be completely filled. This statistic is gathered for units placed into hold as a group - not as individual units.

The field labeled "CURRENTLY ALLOCATED TO PROCESSES:" provides a list of the Processes whose task instances had allocated the Resource at simulation end.

The field labeled "PROCESSES CURRENTLY WAITING:" provides a list of the Process task instances which were suspended while waiting for the Resource at the end of the simulation.

```

SIMULATION TIME =      100.00000 SECONDS
RESOURCE REPORT
RESOURCE          TOTAL  CURRENT... MEAN..... STD DEV... MINIMUM... MAXIMUM...
=====
CPU1
  # IDLE          0.      2.700      2.759      0.      8.000
  REQUEST TIME    0.645      1.291      0.      5.000
  HOLD TIME      30      3.000      1.414      1.000      5.000
  INTO BUSY      110
  OUT OF BUSY    102
  # BUSY          8.000      4.400      2.188      0.      8.000
  BUSY TIME      4.020      2.429      1.000     10.000
  # INACTIVE      0.      0.      0.      0.      0.
  INTO WAIT      22
  OUT OF WAIT    22
  # WAITING       0.      0.300      0.458      0.      1.000
  WAIT TIME      1.384      1.823      0.      5.000
  CURRENTLY ALLOCATED
    TO PROCESSES: PLONG1  PSHRT1  PSHRT1  PSHRT1  PSHRT1
                  PSHRT1
  PROCESSES CURRENTLY
    WAITING: NONE

```

Figure 13-10. Resource Report

13.2.5 Action Report

The Action Report provides the user with statistics on the time consumed by each Action. Statistics are gathered on three aspects of such time consumption, called "useful time", "delay time", and "wasted time".

"Useful time" is equal to the amount of time the Action spent performing the action. "Delay time" is the time between the initiation and completion of an Action during which the execution of the Action (i.e., the Process in which it appears) is suspended. "Wasted time" is applicable to an Action which has an option of "RESTART". If the Action is interrupted and is restarted from the beginning, the time spent performing the Action before it was suspended is "wasted" since it must be redone. Useful time, delay time, and wasted time are calculated only upon the completion of the Action. Therefore, Actions which are active at the end of the simulation are not included in these statistics.

A sample Action Report is shown in figure 13-11. The name immediately below the ACTION heading is the user-defined name of the Action. For the row labeled USEFUL TIME the statistics have the following significance:

TOTAL SAMPLES: the number of times the useful time, delay time, or wasted time was calculated (i.e., the number of times the Action was completed).

MEAN: The average useful time, delay time, or wasted time of this Action over the simulation (i.e., the total time divided by TOTAL SAMPLES).

STD DEV: The standard deviation in the useful time, delay time, or wasted time.

MINIMUM: The minimum useful, delay, and wasted time taken in the execution of the Action over the simulation.

MAXIMUM: The maximum useful, delay, and wasted time taken in the execution of the Action over the simulation.

% TIME OF TOTAL: The percent of the total simulation time for which this Action was executing. Since AISIM allows for the parallel execution of the same Action, this figure can be greater than 100.

Note that % OF TOTAL is not calculated for the delay time or wasted time.

SIMULATION TIME = 100.00000 SECONDS

ACTION REPORT

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
=====	=====	=====	=====	=====	=====	=====
DELAY						
USEFUL TIME	486	2.135	3.280	1.000	28.000	995.000
DELAY TIME	486	0.039	0.357	0.	4.000	
WASTED TIME	0	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
=====	=====	=====	=====	=====	=====	=====
LNGDELAY						
USEFUL TIME	2	4.000	0.	4.000	4.000	8.000
DELAY TIME	2	4.000	0.	4.000	4.000	
WASTED TIME	2	5.000	1.000	4.000	6.000	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
=====	=====	=====	=====	=====	=====	=====
MEDDELAY						
USEFUL TIME	3	4.000	0.	4.000	4.000	12.000
DELAY TIME	3	3.667	0.471	3.000	4.000	
WASTED TIME	3	5.000	1.414	3.000	6.000	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
=====	=====	=====	=====	=====	=====	=====
SRTDELAY						
USEFUL TIME	2	4.000	0.	4.000	4.000	8.000
DELAY TIME	2	4.000	0.	4.000	4.000	
WASTED TIME	2	5.500	0.500	5.000	6.000	

Figure 13-11. Action Report

13.2.6 Queue Report

The Queue Report provides statistics on the utilization of user defined Queues. The report contains information both on the number of entities stored on the Queue as well as information on the impact the utilization of the Queue had on Process execution and suspension. A sample Queue Report is shown in figure 13-12. The rows labeled FILED ON, REMOVED FROM, # IN QUEUE and TIME IN QUEUE key statistics on the manipulation of the Queue itself. The rows labeled TASKS BLOCKED, TASKS RESUMED, # BEING BLOCKED, TIME BLOCKED refer to statistics on Process tasks that have been suspended because they attempted to file an entity on a Queue that was full (i.e., whose maximum number had been exceeded.)

The statistics in each category have the following significance.

The TOTAL NUMBER/FILED ON is the number of entities that have been filed on the Queue over the whole simulation.

The TOTAL NUMBER/REMOVED FROM is the number of entities that have been removed from the Queue over the simulation.

The CURRENT/# IN QUEUE is the number of entities on the Queue at the time of simulation end.

The MEAN/# IN QUEUE is the time weighted average of the number of entities on the Queue over the simulation.

The STD DEV/# IN QUEUE is the standard deviation in the number of entities on the Queue over the simulation.

The MINIMUM/# IN QUEUE is the minimum number of entities on the Queue at any time during the simulation (this statistic is always zero since the Queue will be empty at the start of the simulation).

The MAXIMUM/# IN QUEUE is the maximum number of entities residing on the Queue at any time during the simulation.

The MEAN/TIME IN QUEUE is the average time entities spent on the Queue.

The STD DEV/TIME IN QUEUE is the standard deviation of the in times entities spent on the Queue.

The MINIMUM/TIME IN QUEUE is the least amount of time any entity spent on the Queue.

The MAXIMUM/TIME IN QUEUE is the greatest amount of time any entity spent on the Queue.

The statistics on the blocking of tasks due to the filling of Queues have the following significance.

The TOTAL NUMBER/TASKS BLOCK is the number of Process tasks that were suspended over the simulation due to Queue blocking.

The TOTAL NUMBER/TASKS RESUMED is the number of Process tasks resumed after having been blocked due to the filling of a Queue.

The CURRENT/# BEING BLOCKED is the number of Process tasks blocked at the time of simulation end.

The MEAN/# BEING BLOCKED is the average of the number of Process tasks being blocked over the simulation.

The STD DEV/# BEING BLOCKED is the standard deviation in the number of tasks being blocked over the simulation.

The MINIMUM/# BEING BLOCKED is the fewest number of Process tasks blocked at any time during the simulation.

The MAXIMUM/# BEING BLOCKED is the greatest number of Process tasks blocked at any time during the simulation.

The MEAN/TIME BLOCKED is the average of the times Process tasks were blocked during the simulation.

The STD DEV/TIME BLOCKED is the standard deviation in the times Process tasks were blocked during the simulation.

The MINIMUM/TIME BLOCKED is the least amount of time a Process task was blocked during the simulation.

The MAXIMUM/TIME BLOCKED is the greatest amount of time a Process task was blocked during the simulation.

SIMULATION TIME = 1400.00000 SECONDS

QUEUE REPORT

QUEUE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
=====	=====	=====	=====	=====	=====	=====
BUFFER						
FILED ON	400					
REMOVED FROM	400					
# IN QUEUE		0.	1.343	4.369	0.	24.000
TIME IN QUEUE			4.699	7.185	+3.357E-04	26.097
TASKS BLOCKED	0					
TASKS RESUMED	0					
# BEING BLOCKED		0.	0.	0.	0.	0.
TIME BLOCKED			0.	0.	0.	0.

Figure 13-12. Queue Report

13.2.7 Process Report

This report gives information on all aspects of Process executions. As mentioned before, Processes contend for Resources and many times must wait for another Process to complete before the current Process completes. Times spent in these states as well as other important data are recorded automatically for the user.

The Process Report provides the following statistics:

- 1) TOTAL SAMPLES - the number of times the Process was initiated, the total (overall Process instances) number of times the Process waited for another Process to complete and for required Resources to become available.
- 2) The sum total of time spent in all executions of this Process, sum total of waits on Processes and also Resources.
- 3) The mean time required for execution of the Process, for waiting on Processes, for waiting on Resources.
- 4) The standard deviation of time the Process required for execution, for waiting on Processes, for waiting on Resources.
- 5) The minimum time required for Process execution, minimum time spent waiting for other Processes, minimum time spent waiting for Resources.
- 6) The maximum time required for Process execution, maximum time spent waiting for other Processes, maximum time spent waiting for Resources.
- 7) Total number of times this Process was scheduled to execute.
- 8) The number of times this Process was scheduled to execute by a Load or Scenario.
- 9) The number of times this Process was scheduled to execute due to a call from another Process.
- 10) The total number of times this Process completed execution.
- 11) The total number of times this Process did not complete execution.
- 12) Total number of times the execution of this Process was suspended during execution.
- 13) Names of Items used in this Process.
- 14) Number of each Item created by this Process.
- 15) Number of each Item passed to this Process via the SEND Primitive.

- 16) Number of each Item passed out of this Process via the SEND Primitive.
- 17) Number of each Item destroyed by this Process.
- 18) Total number of each Item used in this Process.
- 19) Mean time each Item was held by this Process.
- 20) Minimum time an Item was held by this Process.
- 21) Maximum time an Item was held by this Process.
- 22) Standard deviation of time an Item was held by this Process.
- 23) Verbal description of the Process.
- 24) How many times each Primitive in the Process was executed.
- 25) Any entry Primitives and their names.
- 26) Names of other Primitives in this Process.
- 27) Any parameters or Items associated with each Primitive in the Process.
- 28) Any comment associated with each Primitive in the Process.

An example of a Process Report is shown in figure 13-13.

PROCESS	TOTAL SAMPLES	SUM.....	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
PROCESS3						
TOTAL	10	0.	0.	0.	0.	0.
PROCESS WAIT	0	0.	0.	0.	0.	0.
RESOURCE WAIT	0	0.	0.	0.	0.	0.

TOTAL # SCHEDULE	# AUTO SCHEDULE	# CALL SCHEDULE	# OF COMPLETE	# NOT COMPLETE	# TIMES SUSPEND.
10	10	0	10	0	0

ITEM	CREATED	RECEIVED	SENT	DESTR'D
ITEM1	0	10	0	10
ITEM2	0	10	0	10
ITEM3	0	10	0	10
ITEM4	0	10	0	10
ITEM5	0	10	0	10
ITEM6	0	10	0	10

ITEM	PROCESS # SMPLS	HOLDING TIME MEAN.....	MINIMUM..	MAXIMUM...	STD DEV...
ITEM1	10	5.00	5.00	5.00	0.
ITEM2	10	4.00	4.00	4.00	0.
ITEM3	10	3.00	3.00	3.00	0.
ITEM4	10	2.00	2.00	2.00	0.
ITEM5	10	1.00	1.00	1.00	0.
ITEM6	10	0.	0.	0.	0.

ITEM	TOTAL # SENT TO PROCESS	-- NUMBER IN CURRENT	RECEIVE POOL -- AVERAGE	MAXIMUM	AVG DELAY TIME DUE TO ITEM	MAX DELAY TIME DUE TO ITEM
ITEM1	10	0	0.50	1.00	0.	0.
ITEM2	10	0	0.40	1.00	1.00	1.00
ITEM3	10	0	0.30	1.00	2.00	2.00
ITEM4	10	0	0.20	1.00	3.00	3.00
ITEM5	10	0	0.10	1.00	4.00	4.00
ITEM6	10	0	0.	1.00	5.00	5.00

PROCESS	DESCRIPTION
PROCESS3	RECEIVE AND DESTROY ITEMS -- SERIAL # NO MATCH

COUNT	ENTRY	OPCODE	PARM	PARM	PARM	COMMENT
10		START	ALL	NO		
10		RECEIVE	ITEM1	ITEM2	ITEM3	
10			ITEM4	ITEM5	ITEM6	
10		DESTROY	ITEM1	ITEM2	ITEM3	DESTROY ALL SIX ITEMS
10			ITEM4	ITEM5	ITEM6	
10		END				

Figure 13-13. Process Report

13.3 COMMANDS RELEVANT TO VIEWING OUTPUT REPORTS

To view output reports of simulation runs of a model from the AISIM READY level, one uses the EDIT command.

Since the output report is too long to fit on a terminal screen, to view it all, one must use some text editing commands. Below is a brief review of the commands that are most useful for this purpose. (This discussion refers to the VAX/VMS EDT text editor).

Note: In the following commands "." represents the current line in the file.

13.3.1 TOP, BOTTOM

To orient the screen to either the top or bottom of the report one should enter one of these two commands.

```
TYPE BEGIN
TYPE END
```

13.3.2 UP, DOWN

To move the report either up or down on the screen n lines issue the command,

```
TYPE .-n
```

or

```
TYPE .+n
```

and the line n lines up or down from the current one will be printed.

13.3.3 FIND

To find a certain sequence of characters, sequence, enter the characters between delimiting single quotes.

```
TYPE 'SEQUENCE'
```

and the screen will print the nearest line down in the text containing the characters sequence.

13.3.4 LIST

To print n consecutive lines down from the one to which one is currently oriented, issues the command,

TYPE ...+n

and the next n lines will be displayed on the screen.

APPENDIX A

OPERATIONAL PROCEDURES AND IMPORTANT INFORMATION

A.1 IMPORTANCE OF DATABASE BACKUP AND ALLOCATION

Processes and the other model entities are stored on disk as they are input to AISIM. Changes and additions made to this information are reflected in the current version of the database on disk. It is possible for this database to be damaged if the computer system fails or if the input session is abnormally terminated while a change or addition is being made so that it is unusable. In addition, errors made in inputting may make the stored information nonsensical if they are severe enough. For these reasons, the BACKUP command is provided.

It is wise to periodically create a backup copy of the database with the AISIM READY level command "BACKUP". Should a database be damaged, it may be recreated from the last BACKUP copy by using the "RESTORE" command.

A.2 ABNORMAL TERMINATION OF A DUI OR AUI SESSION

To terminate a DUI or AUI session normally the user must enter the command END. If the user becomes entwined in a situation which disallows normal system operation, the following procedures should be followed:

It should be noted that while in a DUI session, only the data entered prior to the last SAVE command will remain intact after this procedure is executed. If the system appears to malfunction, caution should be used in issuing a SAVE command. If the database is the source of the malfunction and a SAVE command is issued, the user might destroy the entire database. It is better to lose one session's data (by not saving) than to destroy an entire database.

If the user is on an HP terminal, strike the TERMINAL RESET key until the message "TERMINAL READY" appears in the upper left hand corner of the screen; two strikes in a one-second period are required.

Then on any terminal, type the cntl (control) key and the C key simultaneously.

If no response to these procedures is seen, the user should disconnect the modem, and try to log in and reinitiate AISIM.

If the system responds by displaying "\$" the user should reinvoke AISIM.

A.3 AISIM PLOTS

The following section is intended to describe in detail how the simulation plot results produced by the AISIM Analysis function are generated. This discussion addresses the implementation of the plot function in AISIM with respect to the physical characteristics of the terminal display and the driving software. For a user of AISIM, it is generally not necessary to be aware of implementation specific details. This section has been included because the plot output from AISIM simulation runs is the most visible form of output produced. This data may appear to contradict other results produced by the AISIM Analyze function (output listing statistics). This explanation is intended to describe how this function works so that the AISIM user can explain apparent anomalies.

AISIM produces plotted data for many statistics. The plots represent "instantaneous" output from the simulation because in all cases, a defined statistic is plotted against time (the y-axis is the statistic value, the x-axis is the simulation clock). Time is normally considered to be continuous; therefore, it is "reasonable" to assume that AISIM plots are continuous. In reality, this is not the case. AISIM plots are produced by sampling statistics at discrete intervals during the simulation. Each sample defines a point on the plot. A couple of relationships need to be known to understand how this sampling technique produces plots.

The first relationship a user must be aware of is the resolution of the display screen. The terminal graphics terminals have a raster scan display. A raster is the smallest addressable unit which can be illuminated on the screen. Within the AISIM plot axis there are a fixed number of rasters along the x-axis (700 for the HP terminals, 500 for TEK4105, and 1024 for VT100). What this implies is that up to a fixed number of points can be plotted along the x-axis without exceeding the hardware limitations of the display. When an AISIM user specified a plot be displayed which has more than a fixed number of points, the AISIM software reduces the data sent to the terminal so that it can be displayed. This data reduction has the effect of "ignoring" some points. When points are ignored, the obvious result is that the plots lose accuracy. This can account for discrepancies between the plotted data and the simulation summary results, specifically with respect to the minimum and maximum statistics. The simulation report may indicate that a Resource queue had a maximum length of 100 when a plot of the current number in wait for a Resource over time indicates only a maximum value of 80.

Another problem which can occur with respect to plotting is that the plot sampling can miss activity occurring in the simulation because the sample interval is too long. The following default relationship is embedded in the AISIM software. One hundred data points are sampled for each period in the Scenario definition of a simulation run.

what this implies is that if a Scenario is defined to have only one period, only one hundred plot samples will be collected. The sample interval is calculated as the period length/100.0. Suppose the period length is defined to be 3000 units (where units are seconds, this is 1 hour). Plot samples are collected every 36 units (or 36 seconds). If

activity occurs in the model over time intervals less than 36 units, this data will not be captured for plotting. This could occur if a user wanted to see a plot of disk utilization of a computer system over a one-hour time frame. Since disk operations occur in seconds or less, a plot of the current number busy of the Resource disk would miss most of the data points if samples were taken every 36 seconds.

It is possible to adjust the plot sampling interval in the Scenario definition. The number of samples collected for each plot is computed as the number of periods in the Scenario multiplied by 100 points.

To reiterate, AISIM plots produce graphs of statistics collected during a simulation run, and display the results over time. The data for these plots is collected by sampling discrete intervals. It is not generated by state changes detected by the simulator. Therefore, the "instantaneous" plots of "CURRENT" data over time can disagree with accumulated statistics in the simulation listing.

A.4 PRODUCING HARDCOPIES OF THE TERMINAL DISPLAY

In addition to producing hardcopies of the Process flowcharts, the HP2631G Graphics Printer, the TEK4695 copier, or the HP2623 internal printer can be used to produce hardcopies of the architecture, plots, or Process diagrams.

The user is warned especially against copying forms on the TEK4105 terminal since this action will empty the ink wells on the TEK4695 copier. The interfaces on a TEK4105 terminal define the screen to be a dark blue color, so attempts to copy the forms screen will cause a page full of blue ink.

To produce hardcopies of the terminal display of an HP2647A terminal, the following must be in effect:

- 1) An HP2631G Graphics Printer must be connected to the HP2647A Graphics Terminal with the HP-IB communications bus.
- 2) The HP-IB bus address of the printer must be set to one.
- 3) The printer must be set to ON LINE mode.

To transfer the display information to the printer, the user first presses the <COMMAND> key. This places the terminal in "command mode".

To transfer text (e.g., Plot titles, LISTVAL responses), the user then presses the following keys in succession: <F1> <F1> <F3> <F3> <F3> <F7> <1> <RETURN>.

To transfer graphics (e.g., Architecture displays, plots), the user presses the following keys in succession: <F1> <F1> <F3> <F3> <F4> <F7> <1> <RETURN>.

NOTE: Any text preceding the cursor position will not be transferred. Thus, the user should be sure the cursor is placed in the proper position before placing the terminal in "command mode".

When the transfer process is complete, the user exits the "command mode" by once again pressing the <COMMAND> key.

If the user is on a TEK4105 terminal equipped with a TEK4695 printer, the SCOPY button will copy any data on the screen from the terminal to the printer.

The user can print the smaller size copies by using the following procedure before the copy is made:

1. Press the SETUP key (an asterisk should appear).
2. Type HCSIZE 1
3. Press the SETUP key again
4. Perform the copy

The terminal can be reset for normal copy size by following the above procedure and typing a zero instead of a one in line 2.

If the user is on a HP2623 terminal, the following keys will cause any data on the screen to be copied to the internal printer:

- <modes> - display terminal modes
- <remote> - set terminal off line
- <enter key> - perform copy
- <remote> - set terminal back on line

A.5 RANDOMNESS IN RESULTS

There are ten random number streams available for use by the functions producing the random results associated with Loads, probabilistic branching (with the PROB Primitive), and Action durations.

For the Load entity, the random number stream is used by the probability functions that determine the time between Process triggerings. For the PROB Primitive, the random number stream is used in evaluating whether or not execution should branch to the given point. For the ACTION Primitive, the random number stream is used by the probability functions that determine the duration of an Action.

The user may select the random number stream used by each of these three functions using the EDIT command (see section 7.4) in the AUI. The default values are one, two, three, for Loads, PROB Primitives, and ACTION Primitives, respectively. The current stream assignments can be displayed with the LISTVAL command (see section 7.11) in the AUI.

When simulating a system, the user needs to have a sufficient number of observations to analyze in order to draw valid conclusions. It is sometimes desirable to execute additional simulation runs with the same conditions to obtain additional observations. To do this, the random number streams should be changed for each additional run. Otherwise, the results will not change.

APPENDIX B

AISIM ERRORS

If there are errors detected during the initialization, an error message will be written below the invalid entry. Following is a list of the initialization error messages and their causes.

ERROR - VALUE MUST BE NUMERIC

A non-numeric value was found as the value of a Constant. The defined value of a Constant must be numeric.

ERROR - TABLE ENTRIES MUST BE NUMERIC

A non-numeric value was found as an entry in a D or C type Table. All D or C type Table entries must be numeric.

ERROR - ALPHA TABLE X ENTRY IS ILLEGAL TYPE

In an alpha Table, an x entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - ALPHA TABLE Y ENTRY IS ILLEGAL TYPE

In an alpha Table, a y entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - VARIABLE INITIALIZED TO ILLEGAL TYPE

A Keyword or other illegal type was found as the value of a Variable. Variables must be initialized to Actions, Processes, Queues, Resources, Tables, Alpha Literals, or numerics.

ERROR - ATTRIBUTE DEFINED MORE THAN ONCE

An Item, Process, or Resource attribute was defined more than once. The duplicate attribute definition should be removed.

ERROR - ***** NOT DEFINED AS A GLOBAL CONSTANT

A non-numeric value in the size field of a QUEUE was not defined as a global Constant. A non-numeric value for the size must either be the word "INFINITE" or be a previously defined global Constant.

A non-numeric value in the total or initial units field of a Resource was not defined as a global Constant. The total and initial units of a Resource must each be either a numeric value or be a previously defined global Constant.

In the definition of a Scenario, a non-numeric value in the schedule field was not defined as a global Constant. The schedule must be a numeric value or a defined Constant.

In the definition of a Scenario, a non-numeric value in the priority field was not defined as a global Constant. The priority must be a numeric value or a defined Constant.

ERROR - INITIAL # OF RESOURCE UNITS IS
GREATER THAN TOTAL # OF UNITS

In a Resource definition, the initial number of units defined was greater than the total number of units of that Resource which were to be made available.

ERROR - FROM NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the FROM NODE column was not the name of a defined Resource.

ERROR - TO NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the TO NODE column was not the name of a defined Resource.

ERROR - NEXT NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the NEXT NODE column was not the name of a defined Resource.

ERROR - LINK IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the link specified in the VIA LINK column was not the name of a defined Resource.

ERROR - LABEL MUST START IN COLUMN 1 OR
OPCODE MUST START IN COLUMN 10

In a Process definition, a value was encountered which did not start in column 1 or in column 10. If the value is a label, it must start in column 1, or if it is an opcode, it must start in column 10.

ERROR - OPCODE MUST START IN COLUMN 10

In a process definition, a non-label value was encountered which did not start in column 10. All opcodes must start in column 10.

ERROR - ***** NODE NAME IS NOT RECOGNIZED AS A RESOURCE

An invalid value was encountered in the node field of a Process definition. This field must be blank, contain the word "ALL", or contain a value which resolves to the name of a defined Resource.

ERROR - ***** NAME IN GIVENS LIST IS IN ERROR IN THIS CONTEXT
GLOBAL NAMES, NUMBERS AND CLOCK CANNOT BE GIVEN

The value of a given parameter for the START figure of a Process was either a numeric value or the CLOCK. Numeric values and the CLOCK cannot be defined as given parameters in a Process; local variables must be used.

ERROR - ***** ITEM IN RECEIVES LIST IS IN ERROR

This is a general message indicating an error in a START figure of type "ITEM" of a Process. This message is generally followed by one of the two following messages which more specifically describe the error.

ERROR - ITEM APPEARS TWICE IN RECEIVES LIST

In the definition of a START Primitive of type "Item," an Item was listed more than once. An Item should only occur once in the receives list of the START Primitive.

ERROR - REFERENCE IN RECEIVES LIST IS NOT
DEFINED AS AN ITEM

In the definition of a START Primitive of type "ITEM," a value which was listed in the receives list was not defined as an Item. A Process with an ITEM START can only receive Items.

ERROR - ***** NUMERIC REFERENCE IN CALL PROCESS FIELD

In the definition of a CALL Primitive in a Process, the process name field contained a numeric value or a keyword. This field must contain the name of a defined Process to be initiated.

ERROR - RETURN PARAMETERS NOT ALLOWED FOR CALL NOWAIT OR BLOCK

In the definition of a CALL Primitive in a Process, return parameters were defined, but the CALL option was defined as NOWAIT or BLOCK. Only Processes called with a WAIT option can return parameters.

ERROR - ***** NUMERIC OR GLOBAL MAY NOT BE USED AS RETURN

In the definition of a CALL Primitive in a Process, a numeric value, keyword, or the CLOCK was defined as a return parameter. Numeric values, keywords and the CLOCK cannot be used as return parameters.

ERROR - BRANCH CONTINUATION DOES NOT FOLLOW A BRANCH STATEMENT

In the definition of a BRANCH Primitive of a Process, the label to branch to was not given. A branch Primitive must include a label to branch to.

ERROR - KEYWORD CANNOT BE USED IN PROB

In the definition of a probabilistic BRANCH Primitive of a Process, CLOCK or a keyword was used as the probability of BRANCH. These cannot be used as the BRANCH probability. Valid values for the BRANCH probability are numeric values and local and global Variables and Constants.

ERROR - ***** CHECK REFERENCE MUST BE RESOURCE OR QUEUE

In the definition of a TEST Primitive in a Process, the value to be tested was defined as a numeric, a global Variable, or a global Constant. The value to be tested must be a reference to either a Resource or Queue.

ERROR - ***** NUMERIC REFERENCE INVALID IN RESOURCE FIELD

In the definition of a RESET Primitive in a Process, the value to be reset was a reference to a numeric value. The value to be reset must be a reference to a defined Resource whose allocation is to be changed.

In the definition of an ALLOC Primitive in a Process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to be allocated.

In the definition of a DEALLOC Primitive in a Process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to be deallocated.

ERROR - ***** REFERENCE INVALID IN ALLOCATION TYPE FIELD

In the definition of an ALLOC Primitive in a Process, the value in the allocation type field was invalid. The valid entries are "PARTIAL" and "ALL".

ERROR - BRANCH LABEL ***** NOT DEFINED IN OPD

In the definition of a Process, a BRANCH Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be BRANCHED to.

ERROR - LOOP LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a LOOP Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - CHECK LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a TEST Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - COMPARE LABEL ***** NOT DEFINED IN PROCESS

In the definition of a Process, a COMPARE Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - READ LABEL ***** NOT DEFINED IN OFD

In the definition of a Process, a READ Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - ***** ALREADY DEFINED AS AN ENTRY NAME IN THIS PROCESS

In a Process definition, an ENTRY Primitive was defined twice with the same label. A label can occur only once in a Process.

ERROR - '*****' KEYWORD CANNOT BE ASSIGNED NEW VALUE

In the definition of an ASSIGN Primitive in a Process, an attempt was made to assign a new value to a Keyword other than \$CNODE. Only the \$CNODE keyword can be assigned a new value.

ERROR - NUMERIC QUANTITY CANNOT BE ASSIGNED A VALUE

In an ASSIGN Primitive of a Process, an attempt was made to assign a new value to a numeric value. The only entities which can be assigned a new value are attributes, Variables, and local variables.

ERROR - ***** GLOBAL CONSTANT CANNOT BE ASSIGNED A NEW VALUE

In an ASSIGN Primitive in a Process, an attempt was made to assign a new value to a global Constant. The only entities which can be assigned a new value during a simulation are attributes, Variables, and local variables.

ERROR - "*****" - NOT RECOGNIZED AS A LOGICAL RELATION

In the definition of a COMPARE Primitive in a Process, the relation field was invalid. Valid relations are EQ, NE, GE, GT, LE, and LT.

ERROR - NUMERIC "*****" CANNOT BE ASSIGNED TO

In the definition of an EVAL Primitive in a Process a numeric was specified in the set variable field. The only entities which can be assigned a new value by an EVAL are global variables and local variables.

ERROR - CONSTANT "*****" CANNOT BE ASSIGNED TO

In the definition of an EVAL Primitive in a Process, a Constant was specified in the set variable field. The only entities which can be assigned a new value by an EVAL are global variables and local variables.

ERROR - KEYWORD "*****" CANNOT BE ASSIGNED TO

In the definition of an EVAL Primitive in a Process a Keyword was specified in the set variable field. The only entities which can be assigned a new value by an EVAL are global variables and local variables.

ERROR - ALPHA "*****" CANNOT BE ASSIGNED TO

In the definition of an EVAL Primitive in a Process an alpha literal was specified in the set variable field. The only entities which can be assigned a new value by an EVAL are global variables and local variables.

ERROR - EXPRESSION IS MISSING

In the definition of an EVAL Primitive in a Process no arithmetic expression was specified. Provide an arithmetic expression to be assigned to the set variable.

ERROR - "#" IS AN INVALID CHARACTER IN AN EXPRESSION;
EXPRESSION WILL NOT BE PARSED CORRECTLY.

In the definition of an EVAL Primitive in a Process the arithmetic expression was found to contain the invalid character "#". The valid characters are A through Z, 0 through 9, \$ and _ .

ERROR - "*" IS AN INVALID CHARACTER IN AN EXPRESSION - IGNORED

In the definition of an EVAL Primitive in a Process, the arithmetic expression was found to contain an invalid character. The character will be ignored. The valid characters are A through Z, 0 through 9, \$ and - .

ERROR - EXPRESSION BEGINS IMPROPERLY WITH "***"; A NAME,
NUMBER, FUNCTION, "(", "+", OR "-" IS EXPECTED

In the definition of an EVAL Primitive, an expression within the arithmetic expression was found to begin incorrectly. The expression must begin with a name, a number, a function, an opening parenthesis, a plus sign or a minus sign.

ERROR - EXPRESSION DOES NOT TERMINATE CORRECTLY

In the definition of an EVAL Primitive, an expression within the arithmetic expression was found that did not terminate normally. If the problem was an unmatched left parenthesis, the following line will appear:

- UNMATCHED LEFT PARENTHESIS

ERROR - INVALID CHARACTER "***" FOLLOWS VALID EXPRESSION

In the definition of an EVAL Primitive, a valid expression was followed by a character that is invalid in the current context, either "(", "[", or "]". Check expression syntax.

ERROR - A VALID EXPRESSION IS PRECEDED OR FOLLOWED BY AN INVALID STRUCTURE

In the definition of an EVAL Primitive a partial expression was recognized, but the rest of the expression contained an error which the parser could not identify specifically. Check the syntax of the expression.

ERROR - FUNCTION NAME FOLLOWED BY INVALID CHARACTER "["
- USE PARENTHESES TO ENCLOSE OPERANDS

In the definition of an EVAL Primitive, a term within the arithmetic expression was a function call that was followed by a left bracket instead of a left parenthesis.

ERROR - FUNCTION NAME IS INVALID FOLLOWING LEFT BRACKET
- BRACKETS ARE USED TO ENCLOSE ENTITY ATTRIBUTES

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to contain a function name where an entity attribute was expected. Brackets are used only to enclose attribute names.

ERROR - INVALID CHARACTER "***" FOLLOWING LEFT BRACKET
- BRACKETS ARE USED TO ENCLOSE ENTITY ATTRIBUTES

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to contain an invalid character where an entity attribute was expected. Brackets are used only to enclose attribute names.

ERROR - INVALID CHARACTER "***"
- "]" EXPECTED TO COMPLETE ATTRIBUTE REFERENCE

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to contain an invalid character where a closing bracket was expected. Brackets are used only to enclose attribute names.

ERROR - NUMERIC "*****" ENCOUNTERED FOR ENTITY IN AN ENTITY[ATTRIBUTE] STRUCTURE

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to be a numeric value where an entity name was expected. A Process, Resource, or Item name or a local variable referring to one of these entities would be a valid entity name.

ERROR - GLOBAL VARIABLE/CONSTANT "*****" ENCOUNTERED
FOR ENTITY IN AN ENTITY[ATTRIBUTE] STRUCTURE

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to contain a global Variable or Constant where an entity name was expected. A Process, Resource, or Item name or a local variable referring to one of these entities would be a valid entity name.

ERROR - ALPHANUMERIC VARIABLE "*****" ENCOUNTERED FOR
ENTITY IN AN ENTITY[ATTRIBUTE] STRUCTURE

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to contain an alphanumeric variable where an entity name was expected. A Process, Resource, or Item name or a local variable referring to one of these entities would be a valid entity name.

ERROR - ALPHANUMERIC VARIABLE "*****" ENCOUNTERED IN AN
EXPRESSION

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to be an alphanumeric variable. A term within a numeric expression must evaluate to a number.

WARNING - FUNCTION "*****" REQUIRES ONLY 1 OPERAND -
SECOND OPERAND IS IGNORED

In the definition of an EVAL Primitive, a term within the arithmetic expression was a function call requiring one operand but two operands were supplied. The second operand is being ignored.

WARNING - FUNCTION "*****" REQUIRES ZERO OPERANDS -
OPERANDS ARE IGNORED

In the definition of an EVAL Primitive, a term within the arithmetic expression was a function call requiring no operands but one or two operands were supplied. The operands are being ignored.

ERROR - FUNCTION "*****" REQUIRES TWO OPERANDS

In the definition of an EVAL Primitive, a term within the arithmetic expression was a function call requiring two operands but two were not supplied. Include two operands in the function call.

ERROR - FUNCTION "*****" REQUIRES ONE OPERAND

In the definition of an EVAL Primitive, a term within the arithmetic expression was a function call requiring an operand but none were supplied. Include an operand in the function call.

ERROR - NUMERIC VALUE "*****" ENCOUNTERED WHERE A
TABLE REFERENCE WAS EXPECTED

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to be a numeric value where a Table name was expected. A Table name or a local variable referring to a table entity would be a valid Table reference.

ERROR - KEYWORD "*****" ENCOUNTERED WHERE A TABLE
REFERENCE WAS EXPECTED

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to be a Keyword where a Table name was expected. A Table name or a local variable referring to a Table entity would be a valid table reference.

ERROR - ALPHANUMERIC VARIABLE "*****" ENCOUNTERED
WHERE A TABLE REFERENCE WAS EXPECTED

In the definition of an EVAL Primitive, a term within the arithmetic expression was found to be an alphanumeric variable where a table name was expected. A Table name or a local variable referring to a Table entity would be a valid table reference.

ERROR - ***** NUMERIC REFERENCE INVALID IN PROCESS FIELD

In the definition of a SEND Primitive in a Process, the Process field contained a numeric reference. The Process field must contain a reference of a defined Process.

ERROR - ***** REFERENCE INVALID IN ITEM FIELD

In the definition of a SEND Primitive in a Process, the list of Items to be sent to a Process contained an invalid value. Only Items can be sent to a Process.

In the definition of a CREATE Primitive in a Process, the list of Items to be created included an invalid value. Only Items can be created by a CREATE Primitive.

In the definition of a DESTROY Primitive in a Process, the list of Items to be destroyed included an invalid value. Only Items can be destroyed by a DESTROY Primitive.

In the definition of a FILE Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a reference to a defined Item.

In the definition of a FIND Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a local variable to be set.

In the definition of a REMOVE Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a variable to be set.

ERROR - ***** INVALID QUEUE OPTION

In the definition of a FILE Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a FIND Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a REMOVE Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, and NEXT.

ERROR - ***** REFERENCE INVALID IN QUEUE FIELD

In the definition of a FILE Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue.

In the definition of a FIND Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue, or the name of a valid cross-reference set: Action, Constant, Item, Process, Queue, Resource, Table, or Variable.

In the definition of a REMOVE Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue.

ERROR - ***** - RESUME REFERENCE MUST NOT BE NUMERIC OR GLOBAL

In the definition of the RESUME Primitive, a numeric value or a Constant or Variable was encountered in the Process field. This reference must be a local variable.

ERROR - ***** IS NOT DEFINED AS A FILE NAME

In the definition of a READ Primitive, the file name field contained a name that does not refer to a valid file name. Provide a valid file name.

ERROR - FILE ***** HAS ALREADY BEEN USED IN A WRITE PRIMITIVE

In the definition of a READ Primitive, the file name field contained a file name that is being used to write to. A file can be used either for writing or reading but not both.

ERROR - GLOBAL CONSTANT ***** CANNOT BE ASSIGNED A NEW VALUE

In the definition of a READ Primitive, the variable reference field contained the name of a global constant, which cannot be assigned to during a simulation. A local variable would be valid in the field.

ERROR - NUMERIC QUANTITY ***** CAN NOT BE ASSIGNED A VALUE

In the definition of a READ Primitive, the variable reference field contained a numeric quantity which cannot be assigned to. A local variable would be valid in the field.

ERROR - KEYWORD ***** CAN NOT BE ASSIGNED NEW VALUE

In the definition of a READ Primitive, the variable reference field contained a Keyword which cannot be assigned to. The \$CNODE, \$TASK and \$ITASK Keywords can be assigned to.

ERROR - KEYWORD ***** CANNOT BE MODIFIED, ONLY ATTRIBUTES CAN BE ASSIGNED TO

In the definition of a READ Primitive, the variable reference field was found to contain a Keyword without an attribute. Supply an attribute of the \$TASK or \$ITASK Keyword to be set.

ERROR - ALPHA STRING ***** CAN NOT BE ASSIGNED A VALUE

In the definition of a READ Primitive, the variable reference field was found to contain an alpha literal which cannot be assigned to. A local variable would be a valid set value name.

ERROR - ***** NOT OPENED, TOO MANY EXTERNAL FILES

In the definition of a READ Primitive, the file name field referenced a file and caused the number of files referenced to exceed 78. Only 78 files can be referenced.

In the definition of a WRITE Primitive, the file name field referenced a file and caused the number of files referenced to exceed 78. Only 78 files can be referenced.

ERROR - ATTRIBUTE ***** APPEARS FOR A TYPE THAT CANNOT HAVE ATTRIBUTES

In the definition of a READ Primitive, the variable reference field contained a Keyword with an attribute which cannot be assigned. The \$CNODE Keyword cannot have attributes assigned to.

In the definition of a WRITE Primitive, the variable reference field contained an alphanumeric variable or alpha literal with an attribute which cannot be assigned to.

ERROR - ***** IS NOT DEFINED AS A FILE

In the definition of a WRITE Primitive, the file name field contained a name that does not refer to a valid file name. Provide a valid file name.

ERROR - FILE ***** HAS ALREADY BEEN USED IN A READ PRIMITIVE

In the definition of a WRITE Primitive, the file name field contained a file name that is being used to read from. A file can be used either for writing or reading but not both.

- ERROR - TRACE MODE MUST BE EITHER 'ON' OR 'OFF'

In the definition of a TRACE Primitive, the ON/OFF field contained a value other than "ON" or "OFF". These are the only valid values.

ERROR - ***** IS AN INVALID TIME UNIT

In the definition of the ACTION Primitive, the value in the time units field was not a valid time unit. The valid time units are nseconds, useconds, mseconds, seconds, minutes, hours and days.

In the definition of a Load, the value in the time units field was not a valid time unit. The valid time units are nseconds, useconds, mseconds, seconds, minutes, hours, and days.

In the definition of a Scenario, the value in the time units field for the schedule time was not a valid time unit. The valid time units are nseconds, useconds, mseconds, seconds, minutes, hours, and days.

In the definition of a Scenario, the value in the time units field for the period length was not a valid time unit. The valid time units are nseconds, useconds, mseconds, seconds, minutes, hours, and days.

In the definition of a Scenario, the value in the output time units field was not a valid time unit. The valid time units are nseconds, useconds, mseconds, seconds, minutes, hours and days.

ERROR - ***** IS AN INVALID RESTART OPTION
- USE "RESTART" OR "RESUME"

In the definition of an ACTION Primitive in a Process, the value in the restart field was not a valid ACTION restart option. The valid options are as listed.

ERROR - LOAD NODE IS NOT RECOGNIZED AS A RESOURCE

In the definition of a Load entity, a value was encountered in a node field which was not a reference to a defined Resource. Nodes must be Resources.

ERROR - '*****' IS NOT DEFINED AS A PROCESS

In the definition of a Load, the name specified in the process field was not defined as a Process. The name specified in this field must be a defined Process.

ERROR - ***** IS NOT A LOAD DISTRIBUTION FUNCTION

In the definition of a Load, the name specified in the schedule field was not a valid Load distribution.

ERROR - ***** IS NOT DEFINED AS A CONSTANT OR VARIABLE

In the definition of a Load, a non-numeric value in the rate field was not defined as a global Constant or variable. If the rate field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the mean field was not defined as a global Constant or Variable. If the mean field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the delta field was not defined as a global Constant or Variable. If the delta field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the priority field was not defined as a global Constant or Variable. If the priority field contains a non-numeric value, it must be a defined global Constant or Variable.

ERROR - NO SCENARIO DEFINED

No Scenario was defined. There must be a Scenario defined in order to run a simulation on a model.

ERROR - PERIOD NOT DEFINED

In the definition of a Scenario, the period was not defined. The period length for a Scenario can be a numeric value or a defined Constant.

ERROR - TRIGGER ***** NOT DEFINED AS A LOAD OR PROCESS

In the definition of a Scenario entity, a value in the trigger field was not a Load or Process. Scenario triggers must be either Loads or Processes.

WARNING - "*****" DISTRIBUTION ONLY REQUIRES 1 PARAMETER

In the definition of an ACTION Primitive in a Process, the specified distribution required only one parameter, but two were supplied. The extra parameter should be deleted or the distribution should be changed.

WARNING - **** NOT LEGAL. USING D INSTEAD.

An illegal Table type was specified. The Table is being assumed to be discrete. The valid table types are continuous (c), discrete (d), and alpha (a).

WARNING - ATTRIBUTE INITIAL VALUE IS NOT DEFINED

In the definition of an Item, Process, or Resource an attribute was not assigned an initial value or was assigned an invalid value. Attributes must be initialized.

WARNING - BLANK PRIORITY FIELD ASSUMES PRIORITY 0

In the definition of a CALL Primitive of a Process, the priority field was left blank. The priority is assumed to be zero.

In the definition of a Load entity, the priority field was left blank. The priority is assumed to be zero.

In the definition of a Scenario entity, the priority field was left blank. The priority is assumed to be zero.

WARNING - "*****" IS AN ILLEGAL OPTION. USING NOWAIT INSTEAD.

In the definition of a CALL Primitive of a Process, the option field contained an invalid option; a NOWAIT option is being assumed. The valid options are BLOCK, WAIT, and NOWAIT.

WARNING - "*****" IS NOT RECOGNIZED IN THIS CONTEXT

In the definition of an ASSIGN Primitive of a Process, an attempt was made to assign a numeric value or a Constant or Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

In the definition of an ASSIGN Primitive in a Process, an attempt was made to assign a value to the \$CNODE keyword, or an attempt was made to assign a value to a Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

WARNING - "*****" - NO QUALIFICATION RECOGNIZED FOR IDENTIFICATION

In the definition of a COMPARE Primitive in a Process, an unrecognizable qualifier for a numeric, a global Variable, or a global Constant was encountered. Qualifiers are allowed only for Items, Processes, Resources, and certain keywords.

WARNING - ***** IS NOT AN ACTION DISTRIBUTION - USING CONSTANT

In the definition of an ACTION primitive in a Process, the value in the method field was not a valid Action distribution; the distribution is being assumed to be CONSTANT. The valid distributions are exponent, constant, lognormal, normal, uniform, Weibull, gamma, and Erlang.

**** ERRORS WERE DETECTED DURING MODEL INITIALIZATION

The number of errors specified were encountered during the initialization of the model and prevented the initiation of the simulation.

If an execution error occurs during the simulation, execution will halt and an error message will be printed in the statistical summary. In some cases there may be a Simscript II.5 traceback. This traceback is a hexadecimal formatted report which is to be disregarded by the user. Following the error messages, the statistical summary lists the state of the Process which was executing when the error occurred. The value of all local variables and attached attributes for the Process are listed. All other output reports are also generated.

Following are all of the execution errors which are produced and an explanation of the conditions which cause each error.

EXECUTION ERROR DETECTED IN PROCESS *****

An error occurred in the specified Process which caused an abnormal termination of the simulation.

EXECUTION ERROR - BRANCH PROBABILITY FOR CURRENT
STATEMENT IS NOT A NUMBER

The BRANCH probability in a BRANCH Primitive in a Process does not evaluate to a number.

EXECUTION ERROR - LOOP NUMBER FOR CURRENT
STATEMENT IS NOT A NUMBER

The value of the LOOP counter in a Process is not a number.

EXECUTION ERROR - TEST STATEMENT ENTITY IS
NOT A RESOURCE OR QUEUE

The value to be tested by a TEST Primitive in a Process is not a Resource or a Queue. The TEST Primitive can only test a Resource or a Queue.

EXECUTION ERROR - VALUE OF RESET IN CURRENT
STATEMENT IS NOT A NUMBER

The value for the number of units to be reset by a RESET Primitive is not a number. The value for the number of units to be reset must evaluate to a number.

EXECUTION ERROR - ATTEMPT TO RESET # OF RESOURCE
UNITS OUTSIDE OF LEGAL LIMITS

An attempt was made to reset a number of Resource units which would make the number of units inactive or active greater than the total number of units which were defined for this Resource.

EXECUTION ERROR - VALUE OF UNITS REQUESTED IN CURRENT
STATEMENT IS NOT A NUMBER

The units field in an ALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - VALUE OF PRIORITY IS NOT LEGAL

The Priority field in an ALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - VALUE OF UNITS TO BE RELEASED IN CURRENT
STATEMENT IS NOT A NUMBER

The units field in a DEALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - RESUME ATTEMPTS TO RESUME A PROCESS WHICH
IS NOT SUSPENDED

An attempt was made to resume a Process instance which was not suspended.

EXECUTION ERROR - A REFERENCE IN THE CURRENT PROCESS EVALUATES TO
AN ILLEGAL TYPE FOR THE CURRENT STATEMENT

The Resource field in a RESET Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The Resource field in an ALLOC Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The Resource field in a DEALLOC Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

EXECUTION ERROR - AN ACTION REFERENCE DOES NOT EVALUATE TO A NUMBER

The scheduling time or the scheduling delta time for an action does not evaluate to a number.

EXECUTION ERROR - PRIMITIVE REFERENCE DOES NOT EVALUATE TO AN ACTION

An undefined opcode for a Primitive was encountered. The opcode was assumed to be the name of a reference to an Action, but it did not resolve to a defined Action name.

EXECUTION ERROR - PROCESS IN CURRENT CALL STATEMENT IS NOT
DEFINED AS A PROCESS

An attempt was made by a CALL Primitive to initiate a Process which was not defined. The Process name in a CALL Primitive must be a reference to an entity defined as a Process.

EXECUTION ERROR - PRIORITY IN CALL DOES NOT EVALUATE TO A NUMBER

The priority in a CALL Primitive did not evaluate to a number. The priority for calling a Process must evaluate to a number.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF GIVEN PARAMETERS
BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of given parameters in a CALL Primitive differs from the number of given parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF RETURN PARAMETERS
BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of return parameters in a CALL Primitive differs from the number of return parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - ORDER RELATIONS ARE NOT DEFINED FOR COMPARE TYPES

For the non-numeric types being compared, an invalid relation was specified. The only valid relations for these types is equal or not equal.

EXECUTION ERROR - ILLEGAL ASSIGN: CURRENT NODE
MUST BE SET TO A RESOURCE

An EVAL or ASSIGN Primitive attempted to set the current node to a reference which was not a defined Resource. The current node must be a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A QUALIFIED
TYPE FOR WHICH NO ATTRIBUTE IS DEFINED

An attempt was made to assign a new value to an attribute of an entity for which no attributes can be defined. Only Processes, Resources, and created Items have attributes which can be modified.

EXECUTION ERROR - ***** ATTRIBUTE NOT DEFINED FOR ITEM

An attempt was made to assign a new value to a nonexistent attribute of an Item.

EXECUTION ERROR - ***** ATTRIBUTE NOT DEFINED

An attempt was made to assign a new value to a nonexistent attribute of a Process or a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A TYPE WHICH CANNOT BE MODIFIED

An attempt was made to assign a new value to an entity which cannot be modified; i.e., a global Constant, a number, or a keyword other than \$CNODE.

EXECUTION ERROR - ATTEMPT TO CREATE AN ENTITY WHICH IS NOT AN ITEM

An attempt was made to create an entity which is not an Item. Only references to Items may be in the create list of the CREATE Primitive.

EXECUTION ERROR - ATTEMPT TO DESTROY AN ITEM WHICH IS CURRENTLY FILED ON A QUEUE

An attempt was made to destroy an Item which had been filed on a Queue and not removed before execution of the DESTROY Primitive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO DESTROY AN ITEM WHICH IS NOT DEFINED OR DOES NOT EXIST

An attempt was made to destroy an Item which was not defined or created, or which has already been destroyed.

EXECUTION ERROR - PROCESS FIELD IN SEND STATEMENT IS NOT DEFINED AS A PROCESS

The reference in the Process field of a SEND Primitive was not resolved as a Process. Items can only be sent to a defined Process.

EXECUTION ERROR - ATTEMPT TO SEND AN ITEM WHICH IS CURRENTLY FILED ON A QUEUE

An attempt was made to send an Item to another Process before the Item was removed from the Queue on which it is currently filed.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO SEND A ENTITY WHICH IS NOT DEFINED AS AN ITEM

An attempt was made by a SEND Primitive to send an entity other than an Item to a Process. Only references to Items may be specified in the SEND Primitive.

EXECUTION ERROR - ITEM ***** ATTEMPT TO BE RECEIVED BY PROCESS
***** IS NOT IN PROCESS NEED LIST

An attempt was made to cause a Process to receive an Item which was not on the list of Items which the Process should receive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ENTITY
WHICH CANNOT BE FILED

An attempt was made by a FILE Primitive to file an entity which cannot be filed. Only Items can be filed.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM
ON AN UNDEFINED QUEUE

An attempt was made to FILE an Item on a Queue which was not defined. The queue reference in the FILE primitive must resolve to a defined Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM
WHICH IS ALREADY ON A QUEUE

An attempt was made to refile an Item. An Item can be filed on only one Queue at any given time.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ENTITY
BEFORE OR AFTER AN UNDEFINED ENTITY

An attempt was made to file an entity before or after an entity which did not exist on the Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE
AN ITEM FROM AN UNDEFINED QUEUE

An attempt was made to remove an entity from an undefined Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE
'NEXT' ITEM WHICH DOES NOT EXIST

An attempt was made to remove a non-existent current Item from a Queue.

EXECUTION ERROR - A REFERENCE FOR A QUEUE IN A FIND PRIMITIVE
IS NOT DEFINED AS A QUEUE OR XREF SET

An invalid reference was specified in a FILE Primitive as the queue name. Only Queues or cross reference sets are valid for the Queue name field.

EXECUTION ERROR - ATTEMPT TO SET AN ATTRIBUTE OF AN ENTITY FOR
WHICH NO ATTRIBUTES ARE DEFINED

An attempt was made by a READ Primitive to set an attribute of an entity which has none.

EXECUTION ERROR - ILLEGAL VALUE ***** : CURRENT MODE MUST BE SET
TO A RESOURCE

In a READ Primitive, an attempt was made to assign a value to \$CNODE which was not a Resource. \$CNODE must be set to a defined Resource entity.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO OUTPUT AN INVALID
ENTITY

An attempt was made by the WRITE Primitive to write the name of an invalid or nonexistent entity.

EXECUTION ERROR - ATTRIBUTE ***** APPEARS FOR A TYPE THAT CANNOT
HAVE ATTRIBUTES

An attempt was made by the WRITE Primitive to write the value of an attribute for a type that cannot have attributes. The invalid type is a global variable, a constant or a numeric value.

EXECUTION ERROR - ***** FROM EXTERNAL FILE ISN'T A VALID ENTRY

The value read from a file for a READ Primitive was not the name of a defined entity. All names must be valid entities.

EXECUTION ERROR - ***** ATTRIBUTE OF A RESOURCE IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ***** ATTRIBUTE OF A RESOURCE UNIT NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource unit. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ***** ATTRIBUTE OF A PROCESS IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Process.

EXECUTION ERROR - ***** ATTRIBUTE OF A TASK IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Task.

EXECUTION ERROR - ***** ATTRIBUTE OF QUEUE NOT DEFINED

An attempt was made to reference an invalid attribute of a Queue.
The valid attributes are NQUEUE and TQUEUE.

EXECUTION ERROR - ***** ATTRIBUTE IS NOT DEFINED FOR CURRENT
ITEM REFERENCE ***** IN EXECUTING LOGIC

An attempt was made to reference a non-existent attribute of an Item.

EXECUTION ERROR - ***** ATTRIBUTE SPECIFIED FOR A TYPE
FOR WHICH NO ATTRIBUTES CAN BE DEFINED

An attempt was made to reference an attribute of a type which does
not have attributes. Entities which have attributes are Resources,
Processes, and Items.

EXECUTION ERROR - KEYWORD REFERENCE IS BLANK

When the simulator tried to resolve a Keyword, the reference field
for the parameter was found to be blank.

EXECUTION ERROR - PROCESS NODE HAS NOT BEEN DEFINED

An attempt was made to reference the process node of a Process, but
the node was not defined.

EXECUTION ERROR - REFERENCE FOR \$ NODE IS NOT A PROCESS

When the simulator tried to resolve the Keyword \$NODE, the reference
was not a Process.

EXECUTION ERROR - ROUTE SET ERROR - NO PATH IN NETWORK

When the simulator tried to resolve \$NXTNODE or \$LINK, there was no
valid path defined in the LPT.

EXECUTION ERROR - CNODE FOR EXECUTING PROCESS NOT DEFINED

When the simulator tried to resolve a link or a next node, there was
no current node defined for the executing Process.

ERROR - ILLEGAL TYPE

In a Process which was executing when an abnormal termination of the
simulation occurred, a local variable was of an invalid type.

ERROR - ILLEGAL ATTRIBUTE TYPE

In a Process which was executing when an abnormal termination of the
simulation occurred, a local variable which resolved to an Item, a
Process, or a Resource had an invalid attribute.

EXECUTION ERROR - NON-NUMERIC VALUE ENCOUNTERED FOR OPERAND
IN ***** OPERATION

The operand for the function in an EVAL Primitive does not evaluate to a number.

EXECUTION ERROR - NON-NUMERIC VALUE ENCOUNTERED FOR FIRST
OPERAND IN ***** OPERATION

The first operand for the function in an EVAL Primitive does not evaluate to a number.

EXECUTION ERROR - NON-NUMERIC VALUE ENCOUNTERED FOR SECOND
OPERAND IN ***** OPERATION

The second operand for the function in an EVAL Primitive does not evaluate to a number.

EXECUTION ERROR - ATTEMPT TO PERFORM DIVISION BY ZERO

In an EVAL Primitive, an attempt was made to divide by zero.

EXECUTION ERROR - ATTEMPT TO TAKE THE SQUARE ROOT OF A
NEGATIVE NUMBER

In an EVAL Primitive, the operand of a SQRT function was a negative number.

EXECUTION ERROR - ATTEMPT TO CALCULATE THE ARC COSINE OF
A NUMBER THAT IS NOT BETWEEN -1 AND 1

In an EVAL Primitive, the operand of an arc cosine function was not a valid number.

EXECUTION ERROR - ATTEMPT TO CALCULATE THE ARC SINE OF
A NUMBER THAT IS NOT BETWEEN -1 AND 1

In an EVAL Primitive, the operand of an arc sine function was not a valid number.

EXECUTION ERROR - ATTEMPT TO CALCULATE THE ARC TANGENT OF ZERO

In an EVAL Primitive, an attempt was made to calculate the arc tangent of zero.

EXECUTION ERROR - ZERO OR NEGATIVE NUMBER DETECTED
IN BETA FUNCTION

In an EVAL Primitive, one or both of the operands of a beta function was less than or equal to zero.

EXECUTION ERROR - ATTEMPT TO CALCULATE THE NATURAL LOG OF A
NUMBER THAT IS LESS THAN OR EQUAL TO ZERO

In an EVAL Primitive, the operand of a logarithm function was less than or equal to zero.

EXECUTION ERROR - ATTEMPT TO COMPUTE THE LOG (BASE 10) OF A
NUMBER THAT IS LESS THAN OR EQUAL TO ZERO

In an EVAL Primitive, the operand of a LOG10 function was less than or equal to zero.

EXECUTION ERROR - NAME IN TABLE STRUCTURE IN AN EXPRESSION
DOES NOT EVALUATE TO A TABLE REFERENCE

In an EVAL Primitive, a name followed by parentheses does not evaluate to a Table. This error may also be caused by a misspelled function name.

EXECUTION ERROR - INDEX INTO DISCRETE TABLE
DOES NOT EVALUATE TO A NUMBER

In an EVAL Primitive, an attempt was made to look up a value in a discrete Table, but the index does not evaluate to a number.

EXECUTION ERROR - INDEX INTO CONTINUOUS TABLE
DOES NOT EVALUATE TO A NUMBER

In an EVAL Primitive, an attempt was made to look up a value in a continuous Table, but the index does not evaluate to a number.

WARNING - ALPHA TABLE LOOKUP FAILED

In an EVAL Primitive being used to look up a value in an alpha Table, a value was not found which corresponded to the lookup index value.

WARNING - EMPTY TABLE DETECTED

A Table was encountered which did not have any entries in it.

EXECUTION ERROR - CHKTYP ROUTINE ERROR WITH LINE:

The simulator encountered difficulty in determining the type of a value it was trying to process.

EXECUTION ERROR - INVALID GLOBAL VARIABLE SPECIFIED FOR TXTOAL: *****

The simulator encountered an error during a data conversion operation.

SIMULATOR ERROR IN COMPUTING NEXT TIME ON
METHOD = ***** REF1 = ***** REF2 = ***** STREAM = *****

An error occurred when an attempt was made to compute the next time in the simulation.

SIMULATOR ERROR - QUEUE.START ATTEMPT TO FILE UNSUCCESSFUL

The simulator attempted to restart tasks blocked from a Queue when there were no tasks currently blocked.

SIMULATOR ERROR - ERROR PARSING TERM

A program error occurred while parsing a term within the arithmetic expression of an EVAL Primitive.

SIMULATOR ERROR - ERROR PARSING FACTOR

A program error occurred while parsing a factor within the arithmetic expression of an EVAL Primitive.

SIMULATOR ERROR - ERROR PARSING PRIMARY

A program error occurred while parsing a primary within the arithmetic expression of an EVAL Primitive.

SIMULATION ERROR - ERROR PARSING FUNCTION

A program error occurred while parsing a function call within the arithmetic expression of an EVAL Primitive.

SIMULATOR ERROR - ERROR PARSING IDENTIFIER

A program error occurred while parsing an identifier within the arithmetic expression of an EVAL Primitive.

SIMULATOR ERROR - ATTEMPT TO SUSPEND MORE UNITS
OF ***** THAN EXIST

The simulator attempted to release more Resource units from a Process than the Process had.

SIMULATION ERROR - NONZERO STACK AFTER ENTITY[ATTRIBUTE] EVACUATION

A program error occurred during the processing of an entity[attribute] structure in an EVAL Primitive.

APPENDIX C

GLOSSARY

ACTION - A discrete event that consumes time during a simulation run.

ANALYSIS USER INTERFACE (AUI) - The interface between the user and the AISIM simulator.

ANALYSIS USER INTERFACE (AUI) READY STATE - Any time after the Analysis User Interface has been invoked, except during a simulation period. This state is indicated by the "*" prompt.

ARCHITECTURE DESIGN EDITOR (ADE) - A sublevel of the DUI which provides the user with the graphics commands to construct a system architecture.

ARCHITECTURE DESIGN EDITOR (ADE) MENU - A representation of the valid symbols available to the user during an ADE session for building an architecture. See ADE MENU.

ARCHITECTURE DESIGN EDITOR (ADE) READY STATE - The state of the system while in the ADE that allows the user to enter commands. This state is indicated by the "#" prompt.

ATTRIBUTE - The specific characteristic of a defined entity.

ATTRIBUTE FORM - A list of available attributes from which the user must select one attribute to be used for testing or data sampling.

BLOCK - Used in conjunction with the CALL Primitive (see section 3.9.5) to indicate that the calling task is to call the specified task and wait until all associated tasks are complete before continuing.

BREAKPOINT - A user-specified condition which, when reached, suspends the simulation to allow the user to monitor the current state of the simulation.

CONSTANT - A value that is not subject to change once a simulation run has been started.

DATABASE - The accumulation of data in a specified form related to a specific function or operation.

DEFAULT CONDITION - The condition that exists if no parameters are explicitly stated.

DESIGN USER INTERFACE (DUI) - The interface that allows the user to create or modify a design database.

DESIGN USER INTERFACE (DUI) READY STATE - Any time after invocation of the Design User Interface, except when utilizing the PEI or ADE sublevels of the DUI. This state is indicated by the "*" prompt.

ENTITY - A predefined set of constructs that have user defined attributes (see section 3 for valid AISIM entities). They are the "building blocks" with which the user creates his model.

ENTITY-NAME - The user-defined name of a valid entity.

ENTITY-TYPE - A type as opposed to a specific, user-defined instance of an entity.

FORMS MODE - A specific function that provides areas which may be filled in by the user, and protected fields which define the areas to be filled in.

INFINITE RESOURCES - A feature which allows the simulator to simulate a Process as if there were no limit to the number of Resources available to it.

LOAD - The amount of activity to be applied to the simulation of a process.

L-NODE - A leaf node in an architecture which typically represents an external load on the system.

MODEL - A group of AISIM entities which represent a certain function or group of functions.

NOWAIT - Used in conjunction with the CALL Primitive to indicate that a Process is to be called by a parent Process and the parent Process is to continue processing in parallel.

OFF-SCREEN - The portion of a graphics picture not visible to the user.

ON-SCREEN - The portion of a graphics picture visible to the user.

PERMANENT DATABASE (sometimes referred to as the Design database) - The user-named database, in which the data for a modeled system resides. (As opposed to the working database which temporarily holds Design data while editing that data).

PRIMITIVE - The model entity used to model individual steps in an operation or function. A Process is constructed from a sequence of Primitives.

PROCESS - A graphical representation of a sequence of events, activities and decisions that models a real-world operation or function.

PROCESS EDITOR INTERFACE (PEI) - A sublevel of the DUI that provides the user with the graphics commands to construct Processes.

PROCESS EDITOR INTERFACE (PEI) MENU - A representation of the valid Primitives available to the user during a PEI session.

PROCESS EDITOR INTERFACE (PEI) READY STATE - The state of the system while in the PEI that allows the user to enter commands. This state is indicated by the "#" prompt.

QUERY - A request for information.

RELATIONAL OPERATOR - A set of mnemonics that represent a relation such as equal, not equal, less than, greater than.

RESOURCE - Representations of the real-world objects that are required by a Process to do its work.

REVERSE VIDEO - A feature of a terminal in which dark characters appear on a light background, rather than light characters on a dark background.

SCROLL - The process of moving data displayed on a terminal either up or down.

STATISTICS FORM - A form presented to the user from which one type of statistical value must be selected for use in data sampling.

SYNTAX - A set of rules that determines the structure and arrangement of words and characters.

TRANSLATOR (XLATOR) - The AISIM database translator that translates the Design database into a form suitable for input to the simulator.

VARIABLE - A term whose value is subject to change.

WAIT - Used with the CALL Primitive to indicate that the calling Process is to suspend until the called Process is complete.

WORKING DATABASE - A copy of the user's real database, into which all work is done on a temporary basis.

APPENDIX D

MESSAGE ROUTING SUBMODEL

The message routing submodel provides a means for a user to route messages through a network which is defined by an architecture and a Legal Path Table.

The message routing submodel consists of one Item representing the message dispatched through the system architecture, four Processes representing the activities required for the inter-node communication and other supporting entities. Everything required for this model is included in the AISIM system library and can be merged into a user's model in a simple operation. (See the Library User Interface, section 10).

Although intra-node communication is modeled by means of a collection of four Processes, the user need explicitly invoke with a CALL Primitive (see section 3.9.5) only one of them. To represent the intra-Node triggering of a Process one calls the first Process in the submodel called "MRS". This process is called using a WAIT option if the user wishes to suspend the calling Process until message routing submodel processing is complete. It allows the calling Process to wait for a response message to be sent back to the calling node before it continues processing. If the MRS Process is called with a NOWAIT option, processing in the message routing submodel will proceed concurrently with the calling Process.

The calling process must call process MRS with six GIVEN values; no RETURN values are required. The GIVEN parameters are:

1. the name of the destination process to be triggered,
2. the priority associated with the destination process,
3. the type of message to be generated -- \$REQRESP, \$REQNORE, or \$RESP. \$REQRESP causes a response message to be sent to the origin, \$REQNORE causes no response message to be generated, and \$RESP inhibits both the response message and the triggering of a destination process,
4. the length of the message,
5. the destination node, and
6. the name of the message item.

The user must also set up attributes of the Resources representing the nodes and channels (see section 3.6). All nodes which messages utilize must have an M_ROUTE attribute which gives the nodal processing delay in time units per message. Each channel resource must have a RATE attribute giving a channel transmission delay in time units per character.

The entities that comprise the message routing submodel are described in the following sections.

ITEM MSG

This Item is the basic prototype for messages created by the MRS. If the user does not want specific point-to-point transit times, all statistics for message routing will be accumulated for this one entity. If specific point-to-point transit times are desired, the AISIM user copies MSG to another Item name (through the Design User Interface COPY command described in section 6.1.2) and provides the unique name as parameter six in the call to the MRS Process. All attributes of the Item MSG are essential to the message routing submodel. The attributes are explained below.

DEFINITION OF ATTRIBUTES FOR ITEM MSG:

Attribute Name	Default Value	Description
CNODE	\$CNODE	The current node where the message resides.
FNODE	\$CNODE	The source node of the message.
LENGTH	99999999	The length of the message in bytes.
TYPE	\$REQNORE	The message type. \$REQRESP is a request message requiring a response when the destination is reached. \$REQNORE is a request message with no response required. \$RESP is a response message.
RPROC	\$ERROR	The destination Process name.
RPROCPRI	99999999	The priority for the destination Process.
TNODE	\$CNODE	The destination node.

RESOURCES

No Resources are contained in the message routing submodel. However, to use it, the user must specify an architecture. Each nodal Resource which messages utilize must have an M_ROUTE attribute which gives the nodal processing delay in time units per message. Each channel Resource must have a RATE attribute giving a channel transmission delay in time units per character.

ACTIONS

Two Action entities are used by the message routing submodel -- ROUTE_OH and XFER_OH. ROUTE_OH is found in Process NODEPROC and is used for nodal processing delays while XFER_OH is found in Process CHANPROC for channel transmission delays.

MESSAGE ROUTING SUBMODEL PROCESSES

The message routing submodel contains four Processes. Details of these Processes do not have to be known by the user if the submodel can be used as is. However, if the user needs to make changes, knowledge of how the processes work is essential. This subsection describes the functioning of these Processes.

PROCESS MRS

This is the top-level Process of the message routing submodel and is called when a user wishes to make use of the submodel. It causes a request message to be generated. Following is a list of the parameters of this Process.

PROCESS NAME: MRS -- Generate a request message and pass it to NODEPROC.

LOCATION: Executes in all nodes.

GIVEN: PROCESS (DATA TYPE: PROCESS) -- the name of the process to be initiated in the destination node.

PRIORITY (DATA TYPE: REAL) -- the priority of the destination process.

MSG_TYPE (DATA TYPE: ALPHA) -- the type of message to be created. The only legal values for this parameter are: \$REQNORE -- a request message is created which requires no response, \$REQRESP -- a request message is created which will request a response at its destination, \$RESP -- used only if no process is to be initiated in the destination node and no response is required.

MSG_LNTH (DATA TYPE: REAL) -- the message length.

TO_NODE (DATA TYPE: RESOURCE) -- the message destination node.

MSG (DATA TYPE: ITEM) -- the name of the message item to be created.

RETURN: None

CALLS: NODEPROC

The Process begins by creating a message and initializing various attributes of it. The attributes CNODE and FNODE are initialized to the current node in which the Process is executing. The attribute RPROC is set to the Process that will be triggered at the message destination node. Process will execute. The attribute TYPE is set to the parameter passed in MSG TYPE. The attribute length is set to the length of the message. The destination node is stored in the TNODE attribute. Process MRS then calls Process NODEPROC with a WAIT option and gives it the newly created message. Figure D-1 is a listing of this Process.

```

PROCESS
MNEMONIC      DESCRIPTION
=====
MRS           GENERATE A PROCESS REQUEST MESSAGE AND INITIATE I/O

ENTRY  OPCODE  PARM      PARM      PARM      COMMENT
=====
START  ALL      NO
GIVEN  PROCESS  PRIORITY MSG_TYPE
        MSG_LNTH TO_NODE  MSG
CREATE MSG
ASSIGN MSG_LNTH          CREATE MESSAGE
        MSG              SET MESSAGE LENGTH
        LENGTH
ASSIGN PROCESS          SET PROCESS
        MSG              RPROC
ASSIGN PRIORITY        SET PRIORITY
        MSG              RPROCPRI
ASSIGN TO_NODE         SET DESTINATION
        MSG              TNODE
ASSIGN MSG_TYPE        SET MESSAGE TYPE
        MSG              TYPE
CALL   NODEPROC WAIT    0      EXECUTIVE SERVICING OF MSG
GIVEN MSG
END

LOCAL VARIABLES OF PROCESS MRS
=====
1 PROCESS (X)  2 PRIORITY  3 MSG TYPE  4 MSG_LNTH
5 TO_NODE    6 MSG        (I)  7 NODEPROC (P)

```

Figure D-1. Listing of Process MRS

Process NODEPROC

This Process performs nodal processing and determines whether the message is at its destination. When the Process is called, it is given the message Item. The following describes the parameters of the Process.

PROCESS NAME: NODEPROC -- Nodal Processing

LOCATION: Executes in all nodes.

GIVEN: MSG (DATA TYPE: ITEM) -- This parameter is the name of the message item created in process MRS.

RETURN: None

CALLS: CHANPROC, DESTPROC

The first step of this Process is to assign the name of the current node to a system variable. The processing delay is then calculated and charged against the current node.

The message's current position is compared with its destination. If the message is at its destination, the Process determines whether the message is a request or response message. If it is a request message, the Process DESTPROC is called with a WAIT option and a priority equal to the requested priority. The requested Process is initiated in the destination node by the Process DESTPROC. If the message is a response message, the Process DESTPROC is called to destroy the message. If the message is not at its destination node, the Process CHANPROC is called to forward the message to its next node. Figure D-2 is a listing of this Process.

```

PROCESS
WNEMONIC =====
NODEPROC =====
DESCRIPTION
=====
NODAL PROCESSING AND ROUTING
=====

ENTRY  OPCODE  PARM  PARM  PARM  COMMENT
=====
START  ALL      NO
GIVEN  MSG
ASSIGN  MSG      CNODE      INDICATE CURRENT NODE
        C_NODE
ASSIGN  C_NODE  M_ROUTE  PROCESSING RATE OF NODE
        RT_OVHD
ASSIGN  MSG      LENGTH  GET MESSAGE LENGTH
        MSG LNTH
EVAL    OVERHEAD  COMPUTE PROCESSING DELAY
        MSG LNTH*RT_OVHD
ALLOC   C_NODE  1      ALL  ALLOCATE CURRENT NODE
        $PRIORITY
ROUTE_OH CONSTANT OVERHEAD  DELAY FOR ROUTING
        MSECONDS RESUME
DEALLOC C_NODE  1
COMPARE MSG      CNODE  EQ    RELEASE C NODE TO OTHERS
        MSG      TNODE  CONTROL IS MSG AT DESTINATION?
        CHANPROC WAIT  0      FORWARD MSG TO CHANNEL
CALL    MSG
GIVEN   END      100
BRANCH
ENTRY
CALL    DESTPROC WAIT  0      MESSAGE AT DESTINATION
GIVEN   MSG      CONTEXT SWITCH MESSAGE
ENTRY
END
END

LOCAL VARIABLES OF PROCESS NODEPROC
=====
1 MSG      (I)  2 C_NODE  3 RT_OVHD  4 MSG LNTH
5 OVERHEAD  6 ROUTE_OH (A)  7 CHANPROC (P)  8 DESTPROC (P)

```

Figure D-2. Listing of Process NODEPROC

Process DESTPROC

This Process models the processing of a message at its destination. It terminates request messages, generates response messages and triggers the requested Process. The following describes the parameters of this Process.

PROCESS NAME: DESTPROC -- Destination processing of message items.

LOCATION: Executes in all nodes.

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the message item created in process MRS.

RETURNS: None

CALLS: CHANPROC

This Process determines whether the message is a request or response message. If it is a response message, this Process destroys the message and terminates. If the message is a request message, the name of the requested Process is retrieved from the RPROC attribute of the message, and the process is initiated. DESTPROC waits until the requested process completes. Next, DESTPROC checks the message attribute TYPE to see whether the requesting Process is waiting for a response. If no response is desired, the message is destroyed and DESTPROC terminates. If a response is requested, the message type is changed to response, the destination node is changed to the origin, and the origin is changed to the current node. Then the Process CHANPROC is called to route the message back to its origin. Figure D-3 is a listing of this Process.

```

PROCESS
MNEMONIC
=====
DESTPROC      DESCRIPTION
=====
PROCESSING AT DESTINATION OF MESSAGE

ENTRY  OPCODE  PARM  PARM  PARM  COMMENT
=====
START  ALL      NO
GIVEN  MSG
ASSIGN  MSG      CNODE      CURRENT NODE
        C NODE
COMPARE MSG      TYPE      EQ      IF RESPONSE, DESTROY
        $RESP      DESTROY
ALLOC  C NODE  1      ALL      ALLOCATE CURRENT NODE
        $PRIORITY
ASSIGN  MSG      RPROC      EXECUTE THE CALLED PROCESS
        PROCESS
ASSIGN  MSG      RPROC PRI      SET PRIORITY FOR REQ PROC
        PRIORITY
CALL    PROCESS WAIT      PRIORITY WAIT UNTIL COMPLETE
GIVEN  MSG
RETURN MSG
DEALLOC C NODE  1
COMPARE MSG      TYPE      EQ      DEALLOCATE CURRENT NODE
        $REQNORE      NO RESPONSE REQ -> DESTROY
        $RESP      DESTROY
ASSIGN  MSG      TYPE      CHANGE MSG RESPONSE TYPE
ASSIGN  MSG      FNODE      SWITCH FROM AND TO NODES
        MSG      TNODE
ASSIGN  MSG      CNODE      CURRENT NODE IS FROM NODE
        MSG      FNODE
CALL    CHANPROC WAIT      0      RETURN MESSAGE TO ORIGIN
GIVEN  MSG
BRANCH  END      100
DESTROY ENTRY      TERMINATE MESSAGE AT DEST
        DESTROY  MSG      TERMINATE MSG
END      ENTRY
        END

LOCAL VARIABLES OF PROCESS DESTPROC
=====
1 MSG      (I)      2 C_NODE      3 PROCESS      (X)      4 PRIORITY
5 CHANPROC (P)

```

Figure D-3. Listing of Process DESTPROC

Process CHANPROC

Process CHANPROC extracts the current node and the destination node from the message Item. It accesses the LPT to determine the next node and the connecting channel. The channel is allocated to simulate its use, and the Process NODEPROC is called. The following describes the parameters of process CHANPROC.

PROCESS NAME: CHANPROC -- full and half duplex channel logic

LOCATION: Executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) -- This parameter is the message item created in MRS.

RETURN: None

CALLS: NODEPROC

The first step of this Process is to assign the current node to the system variable \$CNODE and to determine the destination node for the message. Then the next node and next channel are extracted from the LPT, and the channel is allocated. The transfer time for the message is assumed to be a constant rate. The action XFER.OH simulates the time used to traverse the channel. The value of the current-node attribute of the message is changed to the next node to update the message's position, and the system current node indicator (\$CNODE) is also set to the next node. The channel is then deallocated and the Process NODEPROC is called. Figure D-4 is a listing of this PROCESS.


```

PROCESS
MNEMONIC      DESCRIPTION
=====
CHANPROC      FULL AND HALF DUPLEX CHANNEL LOGIC

```

```

ENTRY  OPCODE  PARM      PARM      PARM      COMMENT
=====
START  ALL     NO
GIVEN  MSG
ASSIGN  MSG     CNODE      SET INTERNAL NODE CURRENT
      $CNODE
ASSIGN  MSG     TNODE      GET DESTINATION NODE
      TO NODE
ASSIGN  $NXTNODE TO_NODE    SET NEXT NODE TO DESTN
      NXT NODE
ASSIGN  $CHANNEL TO_NODE    GET CHANNEL TO NEXT NODE
      CHANNEL
ALLOC   CHANNEL 1          ALL  OBTAIN CHANNEL FOR XFER
      $PRIORITY
ASSIGN  CHANNEL RATE      WHAT IS CHANNEL RATE?
      VSPEED
ASSIGN  MSG     LENGTH    MESSAGE LENGTH
      VLENGTH
EVAL    VM_OVHD           CALCULATE TRANSFER TIME
      VSPEED*VLENGTH
XFER_OH CONSTANT VM_OVHD   DELAY DUE TO TRANSFER TIME
      MSECONDS RESUME
ASSIGN  NXT_NODE          MSG RESIDES IN NEXT NODE
      MSG- CNODE
ASSIGN  NXT_NODE          SET INTERNAL NODE REGISTER
      $CNODE
DEALLOC CHANNEL 1          FREE UP CHANNEL AFTER XFER
CALL    NODEPROC WAIT     0    ROUTE MESSAGE TO NEXT NODE
GIVEN  MSG
END

```

```

LOCAL VARIABLES OF PROCESS CHANPROC
=====
1 MSG      (I)      2 TO NODE      3 NXT_NODE      4 CHANNEL
5 VSPEED   6 VLENGTH 7 VM_OVHD     8 XFER_OH      (A)
9 NODEPROC (P)

```

Figure D-4. Listing of Process CHANPROC

END
DATE
FILMED

4-88
DTIC